(Project Proposal) Project 8: Support various parameter distributions like log-uniform in Katib

Link to Google Doc with Comment access: Proposal Link

Profile Information

Name: Shashank Mittal
Resume: Link
Email: latashashank.3@gmail.com
Alternate Email: shashank.mittal.mec22@itbhu.ac.in
Education: Indian Institute of Technology (BHU), Varanasi
Github: shashank-iitbhu
LinkedIn: Shashank Mittal
Slack: Shashank Mittal
Location: Varanasi, India
Typical Working Hours: 9am to 10pm (IST) UTC+5:30

Project Overview

Title: Project 8: Support various parameter distributions like log-uniform in Katib
Skills Required: Kubernetes, Python, Go, YAML
Difficulty: Medium
Length: 350 hours
Mentors: Andrey Velichkevich, Johnu George, Yuan (Terry) Tang, Yuki Iwai
Component: Katib

Project Description

The goal of this project is to enhance the Katib Experiment APIs to support various parameter distributions such as uniform, log-uniform, and qlog-uniform. This improvement aims to align Katib more closely with other hyperparameter tuning frameworks like Hyperopt, which offer a broader range of parameter distributions. Currently, Katib is limited to supporting only uniform distribution for integer, float, and categorical hyperparameters. By introducing additional distributions, Katib will become more flexible and powerful in conducting hyperparameter optimization tasks.

Motivation

In machine learning experiments, different hyperparameters may exhibit different behaviors and sensitivities. As a result, they may require different sampling strategies for efficient exploration of the hyperparameter space. For example, parameters like learning rates are often sampled on a logarithmic scale. Supporting a wider range of parameter distributions in Katib will enable users to adopt more effective tuning strategies, ultimately leading to better-performing models.

Technical Details

Experiment API Changes

The core of this enhancement involves modifying the ***ParameterSpec*** struct to include a new field called Distribution. This field will specify the distribution type for each hyperparameter. The updated struct in ***pkg/apis/controller/common/v1beta1/common_types.go*** will look like this:

```
Unset
type ParameterSpec struct {
    Name          string       `json:"name,omitempty"`
    Distribution  Distribution `json:"distribution,omitempty"`
    FeasibleSpace FeasibleSpace `json:"feasibleSpace,omitempty"`
}

type Distribution string

const (
    UniformDistribution         Distribution = "uniform"
    LogUniformDistribution      Distribution = "logUniform"
    QLogUniformDistribution     Distribution = "qlogUniform"
    CategoricalDistribution     Distribution = "categorical"
)
```

Suggestion Service Logic

For each suggestion service ***(e.g., Optuna, Hyperopt)***, the logic will be updated to handle the new parameter distributions. This involves modifying the conversion functions to map Katib distributions to the corresponding framework-specific distributions.

Optuna

For example, in ***pkg/suggestion/v1beta1/optuna/service.py***, the conversion for Optuna might look like this:

Python

```python
def convert_to_optuna_distribution(parameter_spec):
    if parameter_spec.distribution == "uniform":
        return optuna.distributions.UniformDistribution(
            parameter_spec.feasibleSpace.min, parameter_spec.feasibleSpace.max
        )
    elif parameter_spec.distribution == "logUniform":
        return optuna.distributions.LogUniformDistribution(
            parameter_spec.feasibleSpace.min, parameter_spec.feasibleSpace.max
        )
    elif parameter_spec.distribution == "qlogUniform":
        # Optuna does not have a direct equivalent of qlogUniform, so we need
to
        # implement a custom logic or approximation here.
        raise NotImplementedError("qlogUniform distribution is not yet
implemented for Optuna.")
    elif parameter_spec.distribution == "categorical":
        return optuna.distributions.CategoricalDistribution(
            parameter_spec.feasibleSpace.list
        )
    else:
        raise ValueError(f"Unknown distribution:
{parameter_spec.distribution}")
```

Hyperopt

For Hyperopt, the conversion logic in ***pkg/suggestion/v1beta1/hyperopt/service.py*** might be similar.

Example YAML Definition

Example YAML files will be updated to demonstrate the usage of the new parameter distributions. For example, in ***bayesian-optimization.yaml***:

Unset

```
parameters:
  - name: lr
    distribution: logUniform
    feasibleSpace:
      min: "0.001"
      max: "0.1"
```

Flowchart

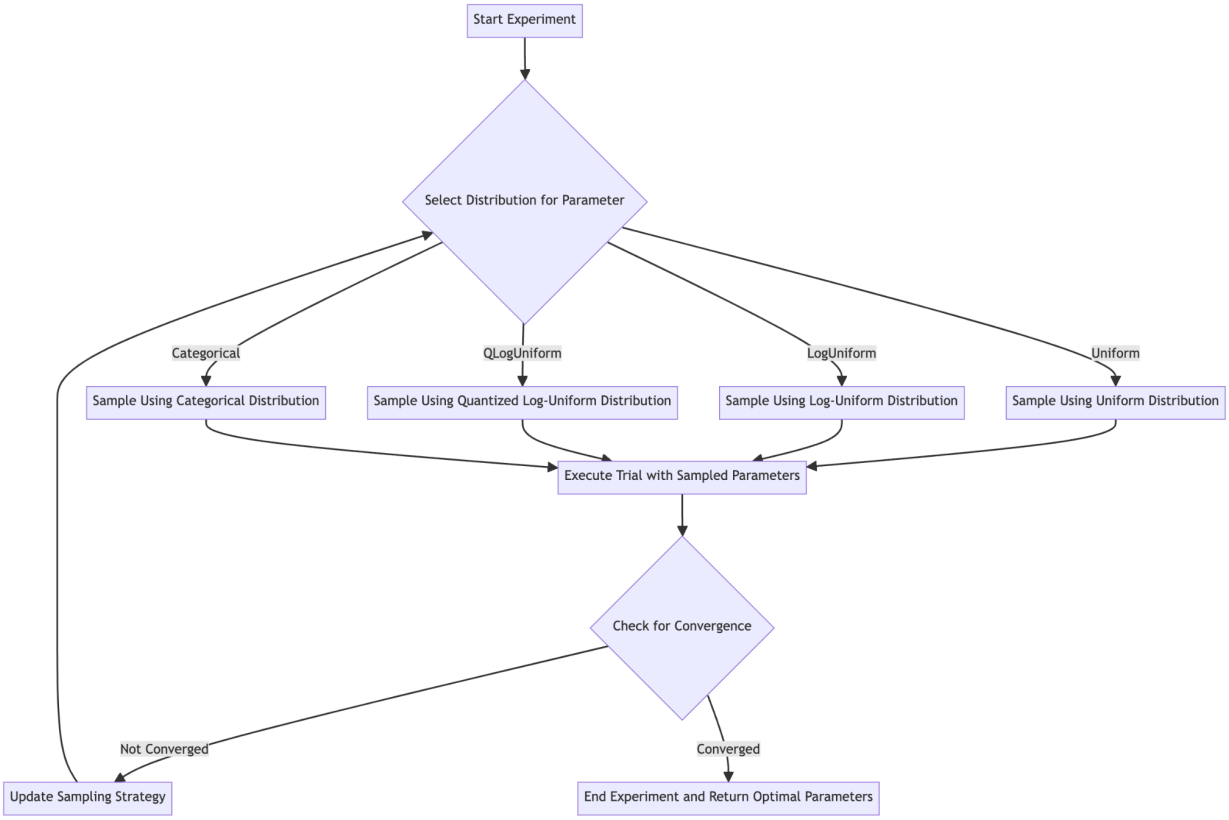

Fig. The flowchart describes the process of conducting a hyperparameter optimization experiment in Katib, involving the selection of parameter distributions, sampling of parameters, execution of trials, and convergence checking to determine the optimal hyperparameters.

Implementation Plan and Timeline

Period	Milestone	Task Description
May 1 to May 26	Community bonding period	- Join community and connect with Mentors - Start working on the feedback for the proposal and start working on Suggestions.
May 27 to June 9	Research and Planning	- Conduct a literature review on hyperparameter tuning and distribution sampling strategies.

		- Explore other frameworks for insights into distribution support.
June 10 to June 23	Update Experiment API	- Modify the ***ParameterSpec*** struct to include the Distribution field. - Update validation logic and API documentation.
June 24 to July 7	Update Suggestion Service for Optuna	- Implement the conversion logic for ***Optuna***. - Write unit tests and update the Optuna suggestion service.
July 8 to July 12	Mid Term Evaluations	- Code Cleanup, Testing and Progress Report/Blog
July 13 to July 21	Update Suggestion Service for Hyperopt	- Implement the conversion logic for ***Hyperopt***. - Write unit tests and update the Hyperopt suggestion service to handle the new distributions.
July 22 to August 4	Update Suggestion Service for Other Algorithms	- Implement and test the conversion logic for other suggestion services (e.g., ***Goptuna***).
August 5 to August 18	Integration and Testing	- Integrate new distributions with the Katib controller and UI. - Conduct end-to-end testing and fix any arising issues.
August 19 to August 25	Documentation and Examples	- Update the Katib documentation with the new parameter distributions.

		- Create and update examples demonstrating the new distributions.
August 26 to September 2	Final Evaluations	- Address any remaining issues or feedback. - Polish code and documentation. - Mentors submit final GSoC contributor evaluations

Previous Contributions to Kubeflow

- Writing Unit Tests for the Katib SDK. Issue #2184
  This is a work in progress, may get merged after the proposal submission deadline.
- Update the developer documentation guide. PR #1995
- Active contribution in bi-weekly community calls and working group calls.

About Me

I am Shashank Mittal, a Sophomore at the Indian Institute of Technology (BHU), Varanasi, pursuing B.Tech in Mechanical Engineering. I am passionate about software development and love to break software.

I was selected as a LFX mentee for the Spring Term 2024 under the LFX Mentorship program facilitated by The Linux Foundation. I was selected to work for KCL language which is a Cloud Native Computing Foundation (CNCF) Sandbox Project.
This requires me to work on the v.0.9.0 of KCL language which requires me to provide Quick Fix features to the existing errors in the language IDE. This project is written in Rust.
Checkout my contributions to KCL in Rust here: Link

Experience in Go

- Constructed a REST API for the LFX Mentorship metrics website using Go. This microservice parses mentorship-related data into PostgreSQL database, generates statistics, and exposes them through a REST API. Link.
- Checkout my contributions in another CNCF project (jaeger-tracing) here: Link

Experience in Python

- Checkout my contributions in Python to Wikimedia Foundation project Scribe-Data here, Link

I have been working with Scribe-Data for a long time now where I set up the whole translation process using a Hugging Face Model.
- Developed a web extension for detecting dark patterns on e-commerce websites for the Dark Patterns Buster Hackathon 2023 organized by the Government of India. Link. This project demonstrates my proficiency in Python and machine learning models such as **XLNet and RoBERTa**. Link

Relevant Skills

- Python
- Go and Rust
- Git and GitHub
- REST APIs
- SQLITE and SPARQL queries.
- Understanding of WikiData
- HTML, CSS, JS and MERN stack.
- CI/CD pipeline
- Docker and Kubernetes

Volunteer Experience

Core Team Member of Club of Programmers (Software Development Group), IIT (BHU) Varanasi

Conclusion

By supporting a wider range of parameter distributions, Katib will significantly enhance its capabilities in hyperparameter tuning. This project will make Katib more compatible with other tuning frameworks and provide users with more flexibility in designing their experiments. The detailed implementation plan and timeline ensure a structured approach to achieving this goal within the specified duration.

Post GSOC

I like this project personally and I love how active and enthusiastic the whole community of contributors is in Kubeflow. This community is something that I want to be part of for a long time and keep contributing to and improving this project even after GSOC and I'll be more than happy to be a part of major changes in this project.