AutoPilotRL

Prof. Dr. Neelima Bayyapu
Preetham M
B L Siddhartha Bhat

HyperPilotRL Idea - 1
Technical Specifications

March 27, 2023

Problem Statement

The HyperParameter Optimization (HPO) Problem can be analytically expressed as a bi-level optimization problem over

$$\max_{\zeta} f(\zeta, \theta^*) \quad \text{s.t.} \quad \theta^* \in \arg\max_{\theta} J(\theta; \zeta)$$

The inner loop finds the best model for the given hyperparameter configuration selected by the outer loop.

The outer loop maximizes over all the hyperparameter configurations.

Summary of Proposed Idea

We extend the framework of **Bayesian Optimization** by modeling dependencies between hyperparameters of the Reinforcement Learning Algorithm using a **Bayesian Neural Network**. We use this model to propose **batches** of parameter values for exploration under **multi-fidelity simplifications**, utilizing the **Hyperband algorithm**.

Technical Details:

1. Programming Languages - Python
2. Frameworks
    a. Stable-Baselines3
    b. Optuna
    c. Ray-Tune
    d. TensorFlow Probability
    e. BoTorch
    f. RLlib
3. Tools
    a. Statistical Analysis Tool for analyzing the logged information during training of different hyperparameter configurations.
    b. Visualization Tools:
        i. Matplotlib
        ii. Seaborn
        iii. Plotly

Completed tasks

1. **Stable-Baselines3 Implementation**
   Extra functionalities added include :-
   a. Saving model at regular intervals.
   b. Saving logs at regular intervals to analyze the learning curve statistics.
   c. Tensorboard integration for real-time visualization of performance metrics.
2. **PyQt5 interface:**
   Graphical User Interface built using PyQt5 for training and loading any of the 1055 available Gym environments.
   a. Train the required environment with customizable RL algorithm, timesteps, and iterations from stable-baselines3 library.
   b. Visualize the performance of the pre-trained environment by rendering it and observing its behavior during sampling.
3. **Custom Wrapper Class for Optuna:**
   Using the Optuna: Hyperparameter Optimization library to build a basic implementation of Hyperparameter Optimization for RL algorithms.

Project Implementation Outline

Single Hyperparameter Configuration Evaluation

1. Setup the information required for Bayesian Optimization algorithm to predict the next optimal configuration to evaluate at.
2. Perform the Bayesian Optimization to infer useful information from previous trials of the objective function.
3. Train the RL Algorithm with the hyperparameters given by Bayesian Optimizer.
4. Evaluate the performance of the model by sampling some episodes on the same environment or a similar environment to test its generalizability power/validation error.
5. Compare the result with the current state-of-the-art performance benchmarks given the fidelities applied(due to less computational power). Estimate its long term performance.
6. Train the Bayesian Neural Network model to capture the dependencies between hyperparameters.

Batch Hyperparameter Configuration Evaluation

1. Setup the information required for Bayesian Optimization algorithm to predict the next batch of optimal configurations to evaluate at.
2. Perform the Bayesian Optimization to infer useful information from previous trials of the objective function.
3. Propose batches of hyperparameter configurations
4. Train the RL Algorithm with the hyperparameters given by Bayesian Optimizer.
5. Evaluate the proposed batches: Evaluate the performance of the model by sampling some episodes on the same environment or a similar environment to test its generalizability power/validation error.
6. Analyze the results: Perform statistical analysis.
7. Train the Bayesian Neural Network model to capture the dependencies between hyperparameters.
8. Compare the result with the current state-of-the-art performance benchmarks given the fidelities applied(due to less computational power). Estimate its long term performance.

Ideas that can be implemented

1. Bayesian Neural Networks

   Bayesian Neural Network architecture can be used to model the dependencies between hyperparameters of the Reinforcement Learning Algorithm.

   Dependencies Modeling: Most of the previously published works use Gaussian Processes(GP) to model the dependency and to fit the objective function. BNNs can be more effective, especially when the relationship between the hyperparameters and the performance of the algorithm is complex and non-linear.

   Scalability: This approach also addresses the limitation of Gaussian Process in handling high-dimensional hyperparameter spaces. The Gaussian Process method requires to compute the inverse of the covariance matrix which is of time complexity $O(n^3)$.

   Noise Tolerance: BNNs can capture heteroscedastic noise and non-Gaussian distribution of the errors. In contrast, the Gaussian Process method of Bayesian Optimization exclusively uses white Gaussian noise to model.

   Transferability: BNNs are more transferable than GPs, as they can learn representations of the hyperparameters that can be used to transfer knowledge from one RL task to another.

   Prior Information: Incorporating prior knowledge is easier in Bayesian Neural Networks compared to Gaussian Processes.

   Interpretability: BNNs can provide more interpretability than GPs, as they can learn representations of the hyperparameters that can be used to understand the effects of different hyperparameters on the performance of the RL algorithm.

2. Distributed Model Selection

   We can use the “Ray” distributed computing framework to parallelize the computation across multiple nodes. This allows us to scale our algorithm to large real-scale projects.

3. Transfer Learning Hyperparameter Configurations

   Transfer learning hyperparameter configurations from a similar environment.
   This requires a similarity comparison metric with the available baseline environments.

4. Analyze the recorded Learning Curve Statistics

   Leverage analysis tools from Statistics to model the behavior of learning curves to better predict the extrapolated learning curve performance of new hyperparameter configurations.

5. Population-Based Training

   Population-Based Training technique is used for Distributed Model Training which involves training multiple instances of a model simultaneously.

6. New Evaluation Metrics

   Relying on a single metric for evaluating the algorithm may not provide a comprehensive assessment. Therefore, it would be beneficial to incorporate a variety of metrics, such as Cumulative Reward, Learning Curve statistics, Exploration-Exploitation Tradeoff, Policy Evaluation.

“Optuna: Hyperparameter Optimization Framework” Structure

Ray Tune: A Research Platform for Distributed Model Selection