AutoPilotRL

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

HyperPilotRL Idea - 1

March 16, 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 the Idea

Extending the framework of Bayesian Optimization and modeling the dependencies between the hyperparameters of the Reinforcement Learning Algorithm using **Bayesian Neural Network** to propose batches of parameter values to explore under Multi-Fidelity simplifications using Hyperband.

Reinforcement Learning Algorithms and their Hyperparameters

1. Deep Q - Learning

    a. Learning Rate
    $\boldsymbol{\alpha} \in [0,\ 1]$

    b. Learning Rate decay schedule
    Decay rate: $\alpha_d \in [0, 1]$
    Decay steps: $n_\alpha \in$ [n_low, n_high]

    c. Discount Factor
    $\gamma \in [0, 1]$

    d. Exploration Rate
    $\boldsymbol{\varepsilon} \in [0, 1]$

    e. Exploration decay rate schedule
    Decay rate: $\boldsymbol{\varepsilon}_d \in [0, 1]$
    Decay steps: $n_\varepsilon \in$ [n_low, n_high]

    f. Network Architecture
    Number of hidden layers: $\in$ [low, high]
    Number of neurons per layer: $\in$ []
    Activation Function: $\in$ {}
    NN Learning Rate: $\in$ []
    Optimizer: $\in$ []
    Regularizer: $\in$ {}
    Batch Size: $\in \{powers\ of\ 2\}$

    g. Initialization of Q-values
    $\in$ [Random Initialization, Pretrained, Kaiming Initialization, Xavier initialization}

2. Proximal Policy Optimization (PPO)

    a. policy

    b. learning_rate

    c. n_steps

    d. Batch_size

    e. n_epochs

    f. gamma

   g. gae_lambda
   h. clip_range
   i. clip_range_vf
   j. normalize_advantage
   k. ent_coef
   l. val_coef
   m. max_grad_norm
   n. ues_sde
   o. sde_sample_freq
   p. target_kl

3. Soft Actor-Critic Algorithm
   a. Replay buffer size
   b. Target entropy
   c. Initial temperature
   d. Learning rate schedule
   e. Batch size
   f. Discount factor
   g. Automatic entropy tuning
4. Advantage Actor-Critic (A2C)
   a. policy
   b. learning_rate
   c. n_steps
   d. gamma
   e. gae_lambda
   f. ent_coef

   g. vf_coef
   h. max_grad_norm
   i. rms_prop_eps
   j. use_rms_prop
   k. use_sde
   l. sde_sample_freq
   m. normalize_advantage
   n.

5. Trust Region Policy Optimization
   a. Maximum KL divergence
   b. Backtracking line search
   c. Value function coefficient, Value function learning rate
   d. Entropy regularization coefficient
   e. Max episode length
6. Twin Delayed Deep Deterministic Policy Gradient
   a. Target policy smoothing coefficient
   b. Target policy update interval
   c. Delayed policy updates
   d. Exploration noise
   e. Replay buffer size
   f. Learning rate

Characteristics of our new Algorithm (draft)

1. **Dependencies between hyperparameters**
   Exploiting the dependencies to guess a better performing hyperparameter configuration.
   For example, if increase in the hyperparameter A usually implies that we have to increase the hyperparameter B to improve the performance, we can then use this

information to reduce the search space by a huge factor.
To observe the dependencies between hyperparameters, we use
the **Bayesian Neural Network** model.

2. **Multi-fidelity**
   A model to probabilistically say that this fidelity has more impact in this environment.
   Expand on this and improve english
3. **Acquisition Function**
   The model that maps the probability distribution over the hyperparameters to the choice of the next hyperparameter.
   This model also leverages all the previous performances over different sets of hyperparameter configurations.
   Draft idea → multiple acquisition functions majority vote? Explore
   USed to balance exploration and exploitation
   Explore on penalized acquisition function from the batch paper
4. **Surrogate Function**
   Provides an approximation to the objective function
   The model takes the Hyperparameters and their corresponding objective function values to predict the objective function value for a new set of hyperparameters.
   Explore the different surrogate models.
   Surrogate functions are used to reduce the number of expensive evaluations needed in hyperparameter optimization, since each evaluating an hyperparameter configuration is expensive, surrogate functions.
   Usually they use GP, see better models.
5. **Employing Batched Computations**
   Simultaneously propose batches of hyperparameter values to explore to find the optimal hyperparameter value leveraging parallelization.
   BOIL algorithm mentioned in the below section can be incorporated.
6. **Vector of evaluation metrics**
   Using a vector of evaluation metrics increases robustness and avoids overfitting to a single metric which may not be optimal in a different environment.
7. **Categorical Search Space**
   The Algorithm must be able to efficiently search through categorical space where there is no relative ordering among the different values the parameter can take.
8. **Hyperparameter Schedules**
   Since it is observed that decreasing some of the hyperparameters over time helps the

model to converge to optima quicker. As we approach close to minima, we would like to make small changes instead of large changes.
This also ensures convergence to the optimal model.

9. **Population based approach**
   This method can be employed along with the Batch computation approach to analyze the performance of different populations with different fidelities.
10. **Hyperband**
    Uses a variant of the HyperBand algorithm to apply multiple levels of fidelity.
11. **Ability to handle Non-stationarity**
    Methods like Dynamic Hyperparameter Tuning, Population-based Training can be used to handle a non-stationary Reinforcement Learning problem.
12. **Using algorithms/techniques which have high early performance**
    FABOLAS very high initial rate of change of performance

Concepts to utilize (draft)

1. Extrapolation of Learning Curves
   To predict the effectiveness of further training the current configuration

2. Need for dependencies between hyperparameters
   Inter-dependencies of hyperparameters can arise due to the complex nature of the problem, where one hyperparameter can affect the performance of another

hyperparameter.

3. An example to illustrate the dependency between the hyperparameters learning rate and discount factor: Consider the scenario where the agent has a high discount factor and a high learning rate. In this case, the agent is more likely to update the Q-values based on recent experiences, which means that it may not fully consider the long-term rewards. This can lead to suboptimal behavior, especially in tasks that require planning for long-term goals.

4. Fidelities
   Fidelities refer to the simplifications or the approximations of the original problem The hyperparameter configurations can be trained on low-fidelity models to estimate its performance on the full problem.

5. Active ensembling - (dont think anyone has done this for HYP-RL optimization)
6. Robustness
   Meta-learning – transfer learning approach
   Ensemble models – makes the algorithm perturbations resistant

7. Objective function modeling
   Spearmint - uses GP, RF, Gradient Boosted Trees, TPE, Sobol Sequential Design
   SMAC - tree based approach to explore the hyperparameter space efficiently
   TPE - One unique characteristic of TPE (Tree-structured Parzen Estimator) implementation is that it uses a tree-structured density model to estimate the distribution of the objective function based on the conditional probability of the hyperparameters

8. Population-based training (PBT) is a technique that dynamically adjusts the hyperparameters of a neural network during training by exploring different hyperparameter configurations and transferring information between them.
9. Complex loss function such as
   Cumulative regret
   Exploration exploitation trade offs (Entropy of the policy)
   Sample efficiency

Stability of learning process

10. Considering the dependency of parameter with at different timestamps
    This is the case when we are dealing with non-stationary RL problems where we need to have dynamic hyperparameters to model the non-stationarity of the problem.

11. Methods to improve sample efficiency

12. The evaluation of the objective function can often take a significant amount of time - to solve this use fidelity - this idea is from Practical BO paper

13. Usage of Expected improvement over time to characterize expected improvement.

14. Modeling the Bayesian Optimization problems with functions whose optimas can be easily found.

15. Suspecting drastic change of performance value as we move in a particular direction using the Bayesian model.

16. Model as a map from different fidelities like training set size to estimated performance.

17. Human experts often study performance on subsets of the data first, to become familiar with its characteristics before gradually increasing the subset size
    Similar techniques can be incorporated in our algorithm.
18. If we have the probability distribution of how changes in 2 parameters changes the output performance, then it is so much easier to do bayesian optimization since the search space is greatly reduced
19. Using similarity between input domain, predict the value at output
20. Just like there is peaks for good regions, model a negative peak for bad regions
21. Use the joint distribution to decide the things.
22. FABOLAS automatically determines the amount of data necessary to (usefully) extrapolate to the full dataset.
23. Using PCA to reduce the dimension space of state space and/or hyperparameter space.

24. After finding the most promising fidelities, set all other fidelities and vary only the most promising ones.
25. Hyperparameter tuning to only consider tuning of parameter which are not a fidelity.
26. New evaluation metrics
    Cumulative Reward
    Learning Curve
    Policy Evaluation
    Value Function Evaluation
    Exploration-Exploitation tradeoff
    Generalization (the ability of the configuration to perform under distribution shifts)
27. Instead of thinking of Bayesian Optimiation as a completely black-box function, we can give it some prior information - which might help improve the search space by a lot.

Ideas from Literature Review (draft)

1. Nguyen et al. 2020 ‘s **BOIL** enhances tuning performance by modeling the training curves, providing a signal to guide search. The algorithm transforms the whole training curve into a numeric score to represent high vs low-performing curves. BOIL introduces a data augmentation technique leveraging immediate information from the training curve to inform the underlying GP model.
   The algorithm also selects the number of epochs that it should evaluate.

2. Daniel, C., Taylor, J., and Nowozin, S. (2016). Learning step size controllers for robust neural network training. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30. → This Automatic framework for learning rate can be improvised and adopted in our project. The paper used variance and gradient of loss function. State representation is used to train policy using REPS algorithm, which

also ensures policy updates to be close to each other by constraining the updates through a bound on KL divergence.

3. “Bayesian Optimization in AlphaGo” by Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, Nando de Freitas (2018)
   Hyperparameters tuned:-

   i. UCT exploration formula
   ii. Node - expansion thresholds
   iii. Hyperparameters associated with distributed implementation of MCTS.
   iv. Hyperparameters of formula for choosing between fast roll-outs and value network evaluation per move.
   v. Formula for deciding search space per move

   Insights on individual contribution of components.
   Acquisition function used - Expected Improvement (EI)

Potential idea :- Reduce cost of function evaluation by decreasing the number of self play games per hyperparameter candidate at the price of higher noise. Guided by this, and central limit theorem for Bernoulli random variables, we choose to evaluate p(θ) with 50 games.

Topics to Explore

1. Bayesian Optimization for High-Dimensional problems
2. Incorporate trust region
3. Auxiliary models
4. Bandits for categorical and acquisition for continuous
5. TURBO - Trust region BO
6. Use kernel design to make categorical to continuous and put it to BNN