Hyperparameter Optimization with Ray Tune

Benno Kaech

DESY - SUSY

November 29, 2021

- Hyperparameters: Parameters which control the Learning Process and are set before the Training
- Examples: Learning Rate, Optimizer and Model Topology etc.
- These Parameters can not be learned with gradient-based Methods
- Algorithms should be robust in Hyperparameters, due to Stochasticity of Learning
  $\rightarrow$ Empirical best Hyperparameter Tuple can be Outlier - can test Neighborhood under Assumption of Lipschitz Continuity

- Grid Search: Exhaustive Search through manually specified Subset of Hyperparameters
- Example: Learning Rate $A = 0.1, 0.01, 0.001$, Network Layers $B = 2, 3, 4, 5 \rightarrow$ Configurations: $x \in A \times B$
- Suffers from Curse of Dimensionality
- Random Search: Outperforms Grid Search, especially with few Hyperparameters
- Example: Learning Rate $\epsilon \sim U(0.1, 0.001)$, where U is Uniform Distribution
- Prior Information can be incorporated with Choice of Distribution
- For integer/categorical Hyperparameters: Rounding of sampled Value

- Sequential Optimization for noisy black-box Functions
- Mostly employed for expensive-to-evaluate Functions
- Should be used for Functions with less than 20 Dimensions [1]
- Objective Function not known → Treat as random Function and place Prior over it (Usually Gaussian Process)
- Update Prior with function Evaluations → Posterior which determines next Hyperparameter Setting
- Drawback: **Sequential** Optimization

[1] A Tutorial on Bayesian Optimization, Peter I. Frazie, ArXiv:1807.02811

- Evolutionary Optimization: Based on biological Concepts of Evolution
    1. Initialization: random Generation of Configurations
    2. Fitness Function: evaluate Configurations on Validation Set
    3. Rank Hyperparameters by Fitness
    4. Replace worst performing Hyperparameters with new Configurations generated by Crossover and Mutation
    5. Repeat 2-4 until no further Improvement
- Population Based Optimization: Multiple Models trained simultaneously, poorly performing Configurations are replaced with better performing Configurations + Noise, keeping the currently learned Model
- Differentiates from Evolutionary-Based by being adaptive during Training

- Built for large Search Spaced of continuous and discrete Hyperparameters, particularly if computational Cost for Evaluation is expensive
- IRACE: Iterated Racing Algorithm, uses statistical Tests to discard poorly performing
- (Asynchronous) Successive Halving ((A)SHA): Begins as random Search but prunes low-performing Models → More Resources for more promising Models. Asynchronous Approach additionally removes Need for synchronous Evaluation of Models
- Hyberband: Invokes SHA or ASHA multiple times with varying Levels of pruning Aggressiveness

- Python Library for Hyperparameter Optimization at any Scale
- Support Pytorch/Keras/XGBoost and many more
- Automatically manages Checkpoints and logging to Tensorboard
- Implementation of previously mentioned Algorithms
- Useable with Cluster Managers (Kubernets, YARN, Slurm, LSF)
- Rich Documentation [2] or personal Github with basic Example for Usage on Slurm [3]