

# Hyperparameter Optimization

with Ray Tune

Benno Kaech  
SUSY ML  
November 29, 2021



# Hyperparameters

- 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  
→ Empirical best Hyperparameter Tuple can be Outlier - can test Neighborhood under Assumption of Lipschitz Continuity

# Grid & Random Search

- Grid Search: Exhaustive Search through manually specified Subset of Hyperparameters
- Example: Learning Rate  $A = 0.1, 0.01, 0.001$ , Network Layers  $B = 2, 3, 4, 5$   
→ 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

# Bayesian Optimization

- 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

---

<sup>1</sup>A Tutorial on Bayesian Optimization, Peter I. Frazier, ArXiv:1807.02811

# Evolutionary & Population Based Optimizations

- 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

# Early Stopping-Based

- Built for large Search Space 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

# Simple Linux Utility for Resource Managment (Slurm)

- **Free** and open-source Job Scheduler
- Key Features:
  - Allocating exclusive or non-exclusive Access to Resources (Computer Nodes) to Users for limited Time
  - Providing Framework for starting, executing, and monitoring Work on allocated Nodes
  - Arbitrating Contention for Resources by managing a Queue of pending Jobs
  - Uses Algorithm to optimize Locality of task Assignments

# Ray Tune

- 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 (Kubernetes, YARN, Slurm, LSF)
- Rich Documentation [\[2\]](#) or personal Github with basic Example for Usage on Slurm [\[3\]](#)