Practical Approaches for Efficient Hyperparameter Optimization

3/16/21 12:59 PM

Speaker: Yonggang Hu (IBM), Xavier Bouthillier (Mila)
Date: 03/16/2021

Typical steps of HPO

1. Designing the search space
2. Choosing the algorithm
3. Running the experiment
4. Visualizing results

*5. Adjust design

A. Search Space Design

Continuous dimensions

- Learning rate loguniform(0.00001, 0.5)
  - Don't avoid large learning rates
- Learning rate schedule
  - SGD, Adam, etc
  - A good prior for gamma of exponential schedule
  - N = loguniform(10, 10000)
  - Gamma = (n-1)/n
- Momentum
  - Affects slightly training time, rarely generalization
- Weight decay (L2 regularization)
  - Loguniform(1e-10, 1e-3)
  - Try large values!
  - When using batch-norm, it turns into some form of adaptive schedule for the learning rate

Discreate dimensions

- Batch size
  - Avoid optimizing it!
  - Crank it up to better use your GPU and keep it fixed
  - Optimal learning rate depends on batch size. Keeping batch-size makes optimization easier
- Size of layers or kernels (CNNs)
  - Loguniform(a,b,discrete=True)
  - Can be tricky!
  - Avoid trying too large sizes, otherwise aggressively early stop training based on running time
- Number of layers
  - Even trickier than size of layers
  - When the architecture needs to be optimized, look into Neural Architecture Search for more efficient solutions

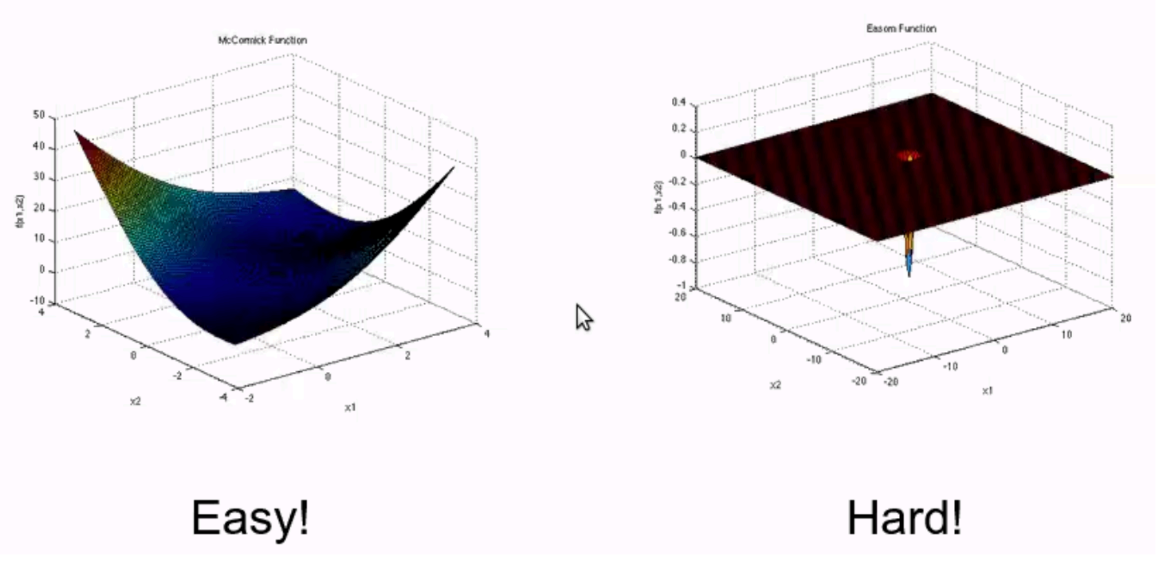
Categorical dimensions

- Type of activation functions
  - Choices(['relu', 'sigmoid', 'tanh', 'swift'])
- Type of optimizer
  - Conditional hyperparameters
  - If only optimize the optimizers' hyperparameter, just optimize separately
- Discretized float, ex: learning rate in choices ([0.001, 0.01, 0.1])
  - Don't do this!
  - Handicaps optimizers and may miss optimal values

B. Choosing the algorithm

Choosing the budget

- Optimization space really matters



General rule of thumbs

1. How many dimensions?
   a. <3: Grid search / Random search
   b. Otherwise, go to next
2. Computation time per trial
   a. < 1m: random search
   b. Otherwise, go to next
3. Late learners
   a. No: Hyperband ASHA
   b. Yes: go to next
4. How many trials>
   a. <200: Bayesian Opt. TPE
   b. >1000: go to next
5. Are all dimensions continuous?
   a. Yes: Bayesian Opt. TPE
   b. No: go to Random Search

Most typical scenarios in Deep Learning are Hyperband ASHA and Bayesian Opt. TPE

C. Running HPO

A few tips:

1. Be careful with small validation sets (e.g. <1000 samples)
   a. Cross-validate, use average over folds as the return value for the HPO algorithm
   b. Randomize your seeds within the cross-validation
2. Protect your script from failures
   a. Handle out-of-memory errors
      i. Client.interrupt_trial()
   b. Handle NaNs during training
      i. Client.report_bad_trial()
3. Mind time
   a. Don't let bad trials continue, use early stopping
   b. Limit training time instead of number of epochs, otherwise some HPs makes training last forever
   c. Make sure there is no bottlenecks in data loaders (ex: use enough loading workers)

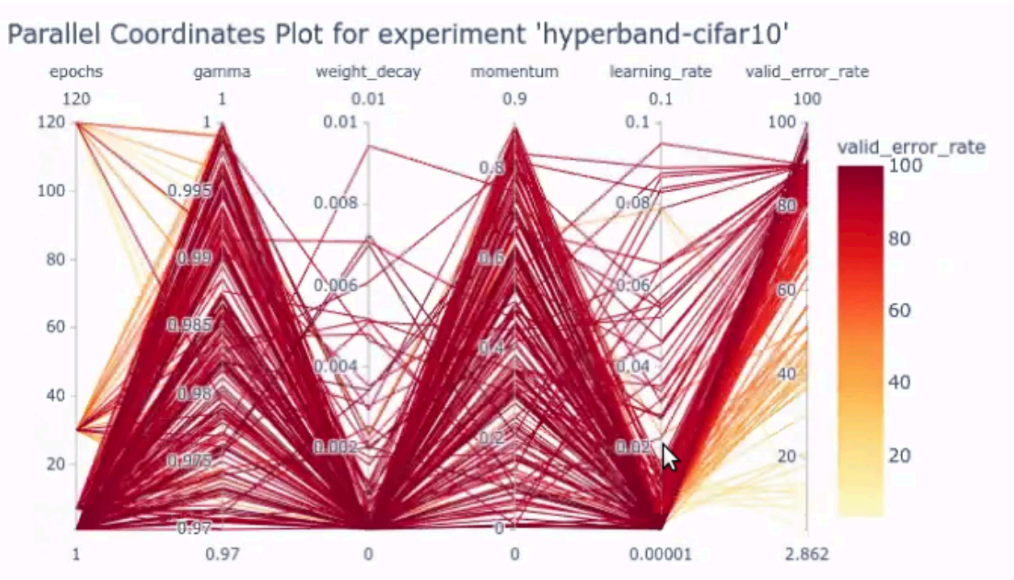
Example with orion:

D. Visualizing results

Convergence

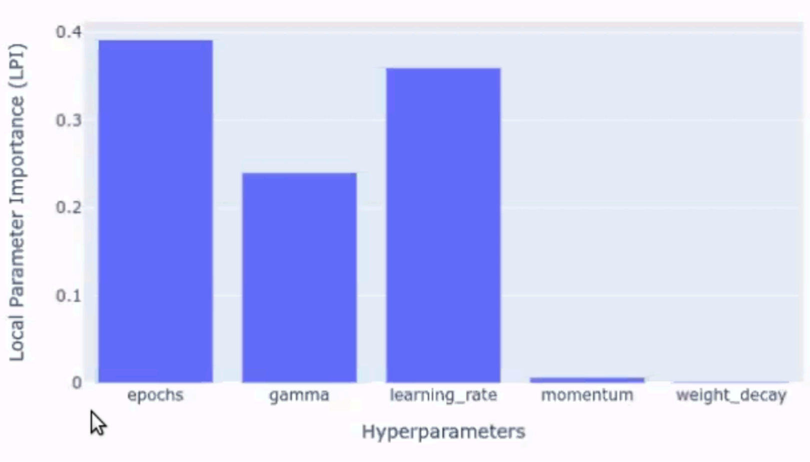
It only tells how quickly it converts, but not necessary on the optimal solution

Parallel coordinates



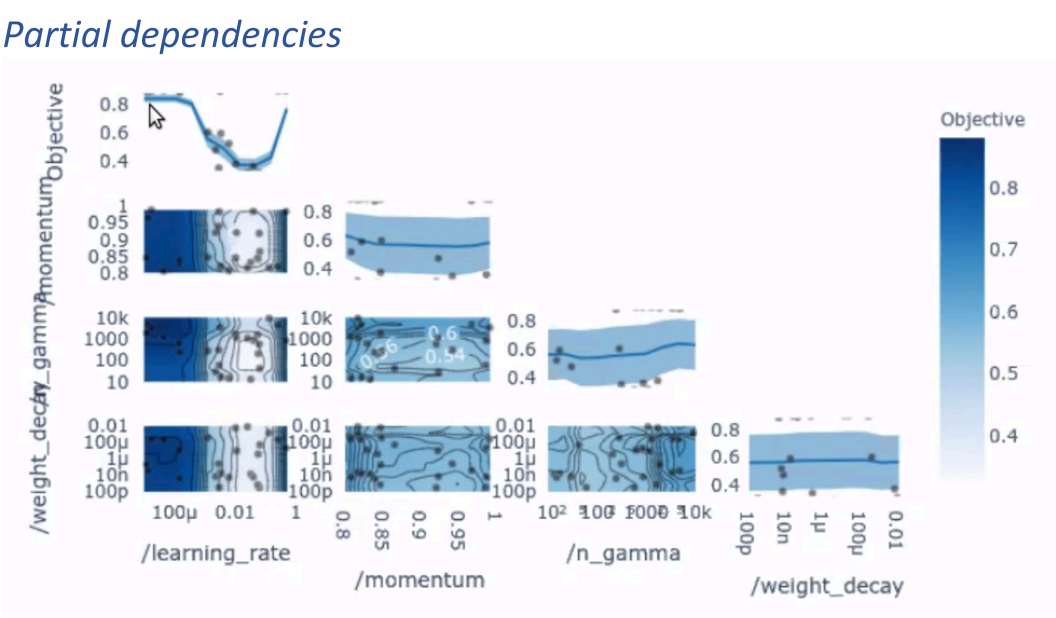
Conveys dense overview in a multi-dimensional space

Local Parameter Importance



Conveys a very compact measure of the importance of the different hyperparameters to achieve the best objective found so far

Partial dependencies



Conveys overview of the search space, what has been explored
Helps identifying best optimal regions of the space

E. Experiment version control

Detects modifications and automatically branch to a new experiment with version tag incremented

Source of changes:

Code
Search Space
Algorithm configuration
Command-line call

Example for tuning learning rate: