Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

Hyperparameter Optimization

Frank Hutter Abhinav Valada

University of Freiburg

Overview of Week 9

1. Hyperparameter Optimization (HPO) in Deep Learning
2. Manual HPO in Deep Learning
3. Blackbox Optimization Methods for HPO
4. Beyond Blackbox Optimization Methods for HPO
5. Hyperparameter Gradient Descent
6. Summary, Further Reading, References

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

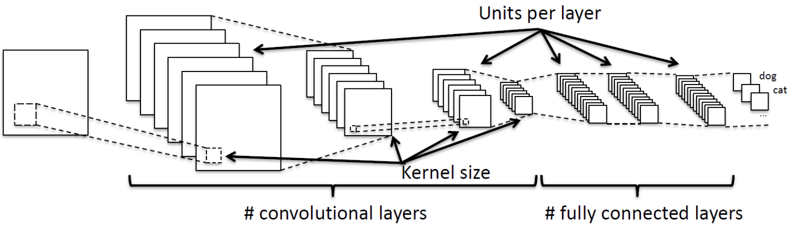
Hyperparameter Optimization (HPO) in Deep Learning

Frank Hutter Abhinav Valada

University of Freiburg

Neural Networks are Very Sensitive to Many Hyperparameters

- Architectural Hyperparameters



- Optimization: SGD variant, learning rate schedule, momentum, batch size, ...
- Regularization: dropout rates, L_2 weight decay, data augmentation, ...

$\leadsto$ Easily 20-50 design decisions

Hyperparameter Optimization: Problem Definition

Hyperparameter Optimization (HPO)

Let

- $\boldsymbol{\theta}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Theta}$,
- $\mathcal{D}_{opt}$ be a training set which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $\mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the loss of $A_{\boldsymbol{\theta}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

The *hyper-parameter optimization (HPO)* problem is to find a hyper-parameter configuration that minimizes this loss:

$$\boldsymbol{\theta}^* \in \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Types of Hyperparameters

- Standard types
  - Numerical
    - Continuous (e.g., learning rate)
    - Integer (e.g., number of layers)
  - Categorical (finite domain, no order)
    - Boolean (special case of binary domain; e.g., dropout on/off)
    - Categorical (general case with domains of more than two options; e.g., choice of optimizer)

- Some hyperparameters θ are only active if other hyperparameters θ' take certain values
  - E.g., weight initialization for layer k is only active if the number of layers is at least k
  - We call these hyperparameters conditional with parents θ'
  - You can always ignore such conditionality, but exploiting it can simplify the problem

Range Transformations of Hyperparameters

- Several hyperparameters naturally lay on a logarithmic scale
  - E.g., a reasonable discretization of learning rates is: $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$
- Good rule of thumb: transform the space such that you'd want to sample uniformly from the transformed space

$$\texttt{log\_learning\_rate} \sim u(-5, 0) \tag{1}$$

$$\texttt{learning\_rate} = 10^{\texttt{log\_learning\_rate}} \tag{2}$$

  - E.g., sampling learning rate from uniform distribution $U[10^{-5}, 10^{-0}]$ yields more than 90% samples greater than $10^{-1} = 0.1$ $\rightarrow$ bad
  - E.g., sampling log learning rate from uniform distribution $U[-5, 0]$ yields 20% samples greater than -1 (i.e., learning rates greater than 10^{-1}) $\rightarrow$ intended

Questions to Answer for Yourself / Discuss with Friends

- Repetition: List the various types a hyperparameter may have.
- Transfer: List some hyperparameters other than the learning rate for which you would like to apply a log transformation.
- Transfer: Think of a hyperparameter for which you would like to use a transformation other than the log transform.

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

Manual HPO in Deep Learning

Frank Hutter Abhinav Valada

University of Freiburg

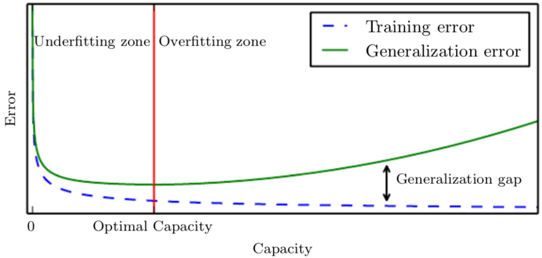
Motivation for Manual Hyperparameter Tuning

- Full understanding of DL methods also requires understanding their hyperparameters
  - Effects to take into consideration
    - training error
    - generalization error
    - time complexity (forward/backward passes)
    - memory requirements
  - E.g., what effect does it have to
    - reduce the L_2 regularization strength?
    - increase the network's depth?
- Manual tuning by experts can be more sample-efficient than automated HPO tools
- Hands-on knowledge also helps to use automated HPO tools effectively
  - Which hyperparameters should you tune?
  - Which fixed values should you use for the others?
  - Which ranges should you consider for each hyperparameter to be tuned?

Classical View on Generalization Error as a Function of Capacity

Generalization error for single hyperparameters typically follows a U-shaped curve

- low capacity: high training error, low generalization gap (underfitting)
- high capacity: low training error, high generalization gap (overfitting)
- optimal model capacity: lowest generalization error

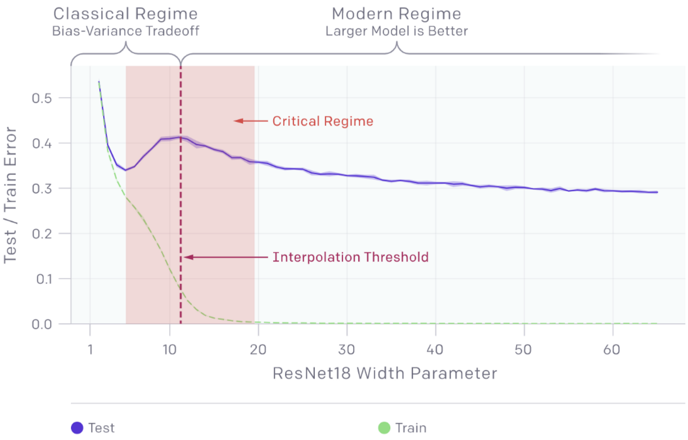


Effective capacity: depends on representational capacity & ability to use this

[Image source: Goodfellow et al., 2016 (Figure 5.3)]

The Deep Double Descent Hypothesis

- There is evidence that generalization error, as a function of effective capacity
  - first decreases
  - then increases
  - then decreases again



- Critical regime is where model transitions from underparameterized to overparameterized
- Error peak often around where training error just reaches (close-to) zero

[Image source: Nakkiran et al, 2019]

Measures Against Underfitting and Overfitting

- Training set performance is poor (underfitting):
  - ⇝ The model's effective capacity is too small; check the following:
    - representational capacity (number of layers, etc)
    - optimization algorithm's ability to minimize the cost function
    - degree of regularization by training procedure and cost function
  - ⇝ Add more layers and more hidden units to each layer
  - ⇝ Tune the optimizer's hyperparameters (especially learning rate)

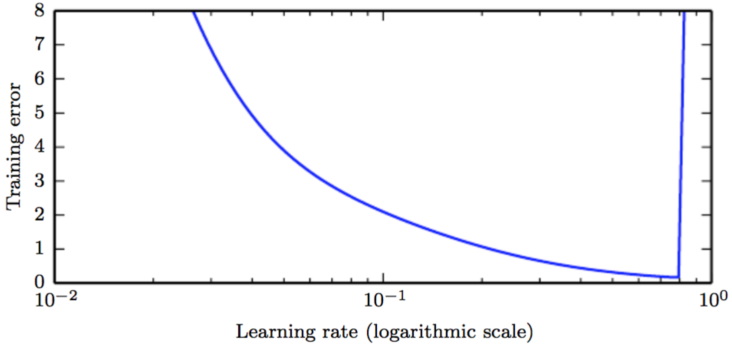
- Performance good on training set, poor on validation set (overfitting):
  - ⇝ Assess different ways of reducing the model's effective capacity
    - reduce model size or try different types of regularization
    - double descent phenomenon: might also help to increase model size
  - ⇝ Can also be due to software bug (model saving, different preprocessing)
  - ⇝ Very effective if possible: more data!
    - plot train & test performance as a function of training set size to judge how promising this is

Practical Approach against Underfitting and Overfitting

- First, make sure you can fit your training data well
  - Don't worry about overfitting at this stage
  - If you can't even get your training error down, you'll also have poor test performance
  - You can use a small subset of the data at this stage, so that it doesn't cost much time
  - Most optimization hyperparameters can be set at this stage, only based on training loss
- Then, combat overfitting
  - Assess how the model reacts to larger subsets of data
  - Monitor both training and validation performance
  - Play with different regularizations and model size
  - The regularization hyperparameters can only be set at this stage
- In all of this, the learning rate is very important
  - The learning rate is typically the most important hyperparameter
  - Whenever you change some hyperparameters check whether it should also be updated

Special Consideration: Learning Rate

- The most important hyperparameter in deep learning
- Controls effective capacity of the model in a more complicated way
    - Quite extreme U-shape
- Simple heuristic: increase until training diverges, then reduce it a bit



[Image source: Goodfellow et al., 2016 (Figure 11.1)]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How can you increase the capacity of your neural network?
- Repetition: What should you do if you experience underfitting?
- Repetition: What should you do if you experience overfitting?
- Transfer: How does switching on dropout affect these characteristics?
  - training error
  - generalization error
  - time complexity (forward/backward passes)
  - memory requirements

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

Blackbox Optimization Methods for HPO

Frank Hutter Abhinav Valada

University of Freiburg

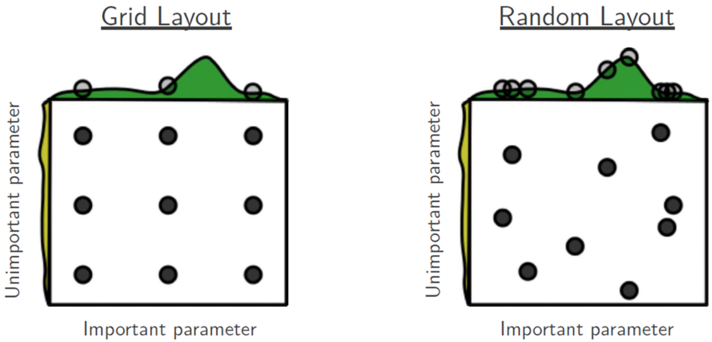
Blackbox Hyperparameter Optimization

Hyperparameter setting $\boldsymbol{\theta}$ $\rightarrow$ $\rightarrow$ Validation error f($\boldsymbol{\theta}$)

- Only mode of interaction with f: querying f's value at a given $\boldsymbol{\theta}$
- Blackbox optimization: find $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} f(\boldsymbol{\theta})$
- Blackbox optimization for HPO: define $f(\boldsymbol{\theta}) := \mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$
- Function may not be available in closed form, not differentiable, noisy, etc.

Blackbox HPO Method 1: Random Search

- Select configurations uniformly at random (completely uninformed)
- Global search, won't get stuck in a local region
- Parallelizes nicely and is at least better than grid search:



[Image source: Bergstra et al, JMLR 2012]

Blackbox HPO Method 2: Local Search

(also sometimes jokingly called "Graduate Student Descent")

Start with some configuration $\boldsymbol{\theta}$
repeat
  Modify a single hyperparameter
  if *results on benchmark set improve* **then**
    keep new configuration
until *no more improvement possible (or "good enough")*

$\leadsto$ Manually-executed first-improvement local search

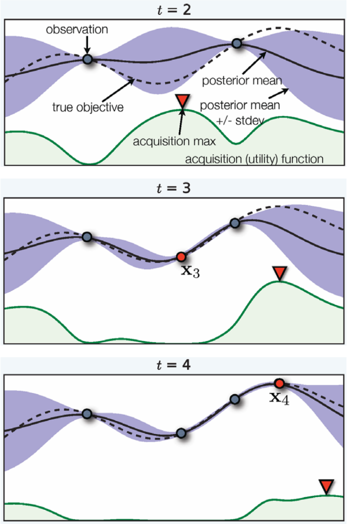
Blackbox HPO Method 3: Bayesian Optimization

General approach

- Fit a probabilistic model to the collected function samples $\langle \boldsymbol{\theta}, f(\boldsymbol{\theta}) \rangle$
- Use the model to guide optimization, trading off exploration *vs* exploitation

Popular approach in the statistics literature since [Mockus, 1978]

- Efficient in # function evaluations
- Works when objective is nonconvex, noisy, has unknown derivatives, etc
- Recent convergence results
  [Srinivas et al. 2009; Bull et al. 2011; de Freitas et al. 2012; Kawaguchi et al. 2015]



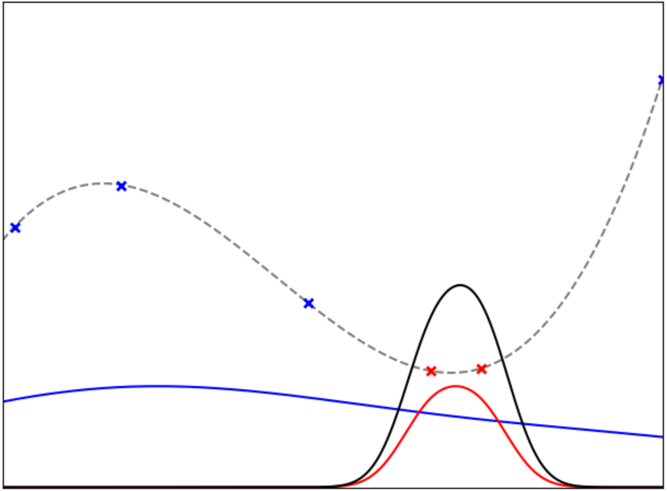
[Image source: Brochu et al, 2010]

Variant: Tree of Parzen Estimators (TPE) [Bergstra et al., 2011]

- non-parametric KDE for $p(\boldsymbol{\theta}$ "is good"$)$ instead of Gaussian Processes modelling $p(f(\boldsymbol{\theta})|\boldsymbol{\theta})$
- equivalent to expected improvement

+ efficient: $\mathcal{O}(N \cdot d)$
+ complex search spaces with priors
+ parallelizable

- not as sample efficient as GPs

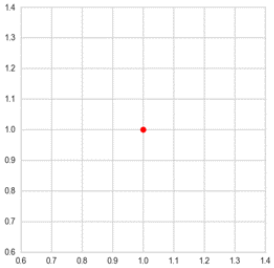


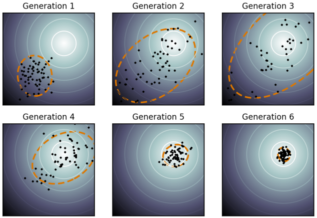
Blackbox HPO Method 4: Population-based Methods

- Population of configurations
  - Maintain diversity
  - Improve fitness of population

- E.g., evolutionary strategies [Beyer & Schwefel, 2002]

- Popular variant: CMA-ES [Hansen, 2016]
  - Very competitive for HPO of deep neural nets [Loshchilov & Hutter, 2016]
  - Embarassingly parallel
  - Purely continuous

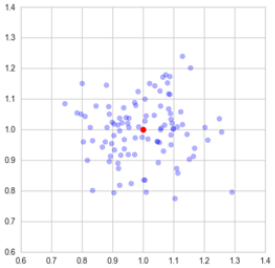


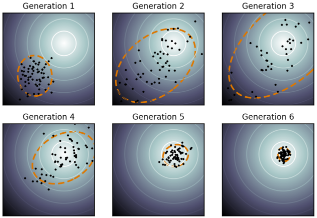


[Image source: Wikimedia Commons]

Blackbox HPO Method 4: Population-based Methods

- Population of configurations
  - Maintain diversity
  - Improve fitness of population
- E.g., evolutionary strategies [Beyer & Schwefel, 2002]
- Popular variant: CMA-ES [Hansen, 2016]
  - Very competitive for HPO of deep neural nets [Loshchilov & Hutter, 2016]
  - Embarassingly parallel
  - Purely continuous

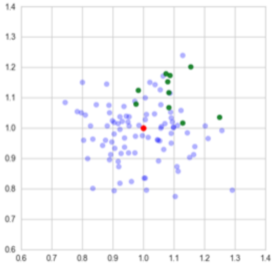


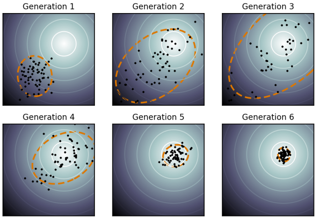


[Image source: Wikimedia Commons]

Blackbox HPO Method 4: Population-based Methods

- Population of configurations
  - Maintain diversity
  - Improve fitness of population
- E.g., evolutionary strategies [Beyer & Schwefel, 2002]
- Popular variant: CMA-ES [Hansen, 2016]
  - Very competitive for HPO of deep neural nets [Loshchilov & Hutter, 2016]
  - Embarassingly parallel
  - Purely continuous

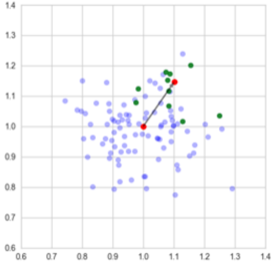


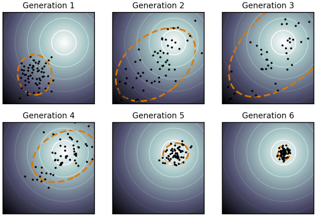


[Image source: Wikimedia Commons]

Blackbox HPO Method 4: Population-based Methods

- Population of configurations
  - Maintain diversity
  - Improve fitness of population
- E.g., evolutionary strategies [Beyer & Schwefel, 2002]
- Popular variant: CMA-ES [Hansen, 2016]
  - Very competitive for HPO of deep neural nets [Loshchilov & Hutter, 2016]
  - Embarassingly parallel
  - Purely continuous





[Image source: Wikimedia Commons]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: List four blackbox HPO methods other than grid search
- Repetition: Explain the main principles of Bayesian optimization

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

Beyond Blackbox Optimization Methods for HPO

Frank Hutter Abhinav Valada

University of Freiburg

Speedup Techniques for Blackbox HPO

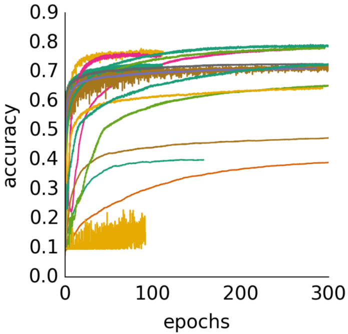
- Meta-learning across datasets
- Extrapolation of learning curves
- Multi-fidelity optimization

HPO Speedup Technique 1: Meta-learning across datatsets

- Meta-Learning: learning about learning methods
  - Learn (and optimize) the performance of learning methods based on data
  - Generate new learning methods from scratch
  - Learn to transfer knowledge across tasks and domains
- Lots of work on meta-learning for HPO
  - Learn across datasets [Swersky et al, 2013; Bardenet et al, 2013; Yogatama et al, 2014; Perrone et al, 2018; Feurer et al, 2018]
  - Inititalize hyperparameters to values that worked well on previous datasets [Feurer et al, 2015]

HPO Speedup Technique 2: Extrapolation of learning curves

- Humans can predict learning curves from their prefix
  - Terminate learning curves that are not promising



Exemplary learning curves of deep neural networks

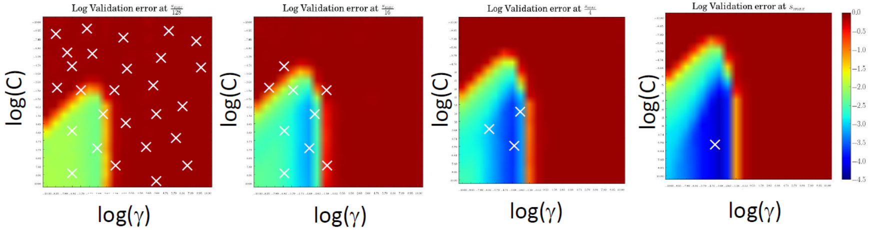
- We can learn to make these predictions from data
  [e.g., Domhan et al, 2014, Gargiani et al, 2019]

[Image source: Domhan et al. 2014]

HPO Speedup Technique 3: Multi-Fidelity Optimization

Multi-Fidelity Optimization: use Cheap Proxies of Expensive Blackbox to Speed up Search

- If we use iterative ML algorithms
  - We can stop poor runs early
- If we use k-fold cross-validation
  - We can reject poor hyperparameter settings after few folds
- If runs on smaller datasets are faster (almost always)
  - We can quickly rule out bad models based on data subsets
  - Example: SVM on MNIST: up to 1.000-fold speedups [Klein et al, AISTATS 2017]

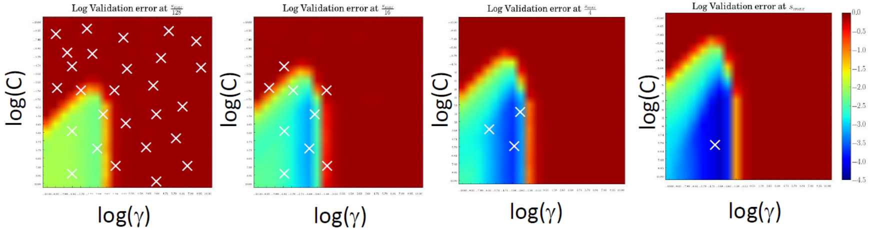


[Image source: Klein et al. 2017]

HPO Speedup Technique 3: Multi-Fidelity Optimization

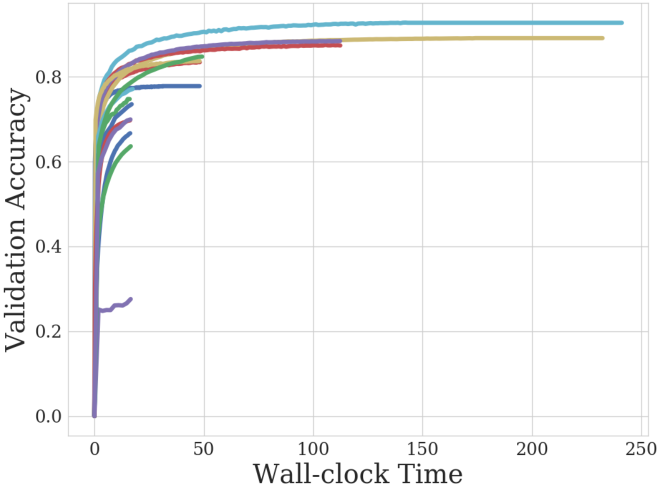
- One can model performance as a function of fidelity and configuration, and then choose them jointly
  [e.g., Swersky et al, 2013; Klein et al, 2017; Kandasamy et al, 2016; Kandsamy et al, 2017; Wu et al, 2019; Takeno et al, 2019]

- Simpler algorithms: successive halving [Jamieson and Talwalkar, AISTATS 2016] and Hyperband [Li et al, ICLR 2018]

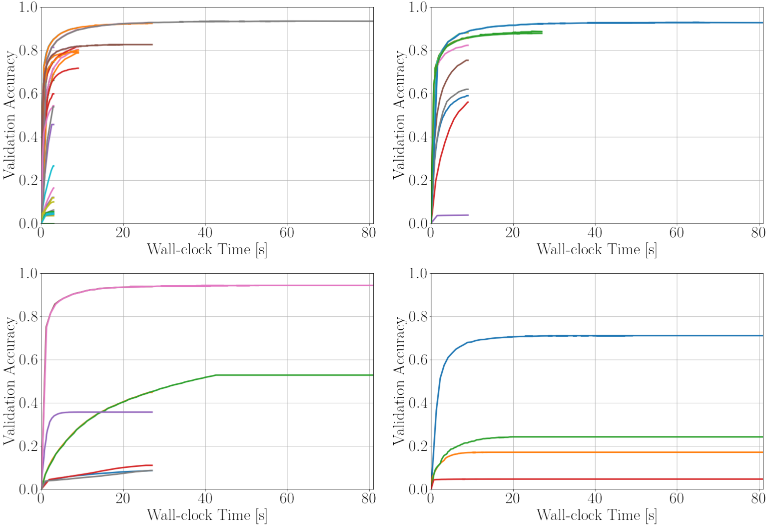


[Image source: Klein et al. 2017]

Successive Halving with a Wall Clock Time Budget



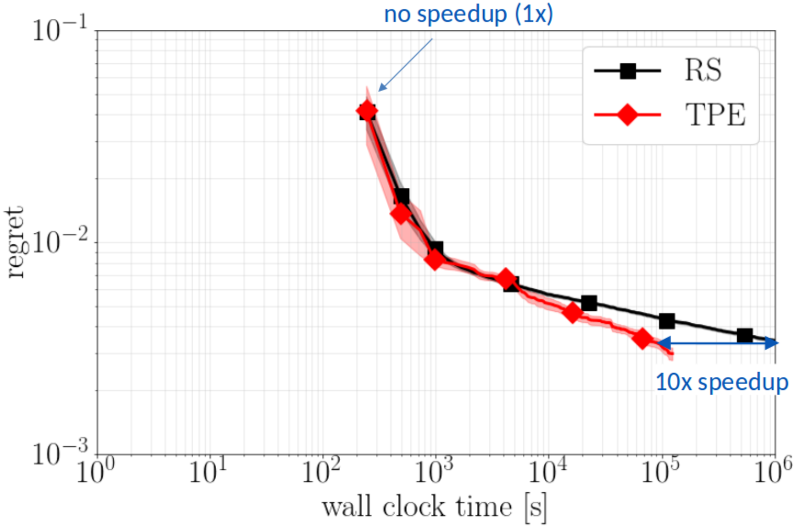
Hyperband with a Wall Clock Time Budget: 4 iterations

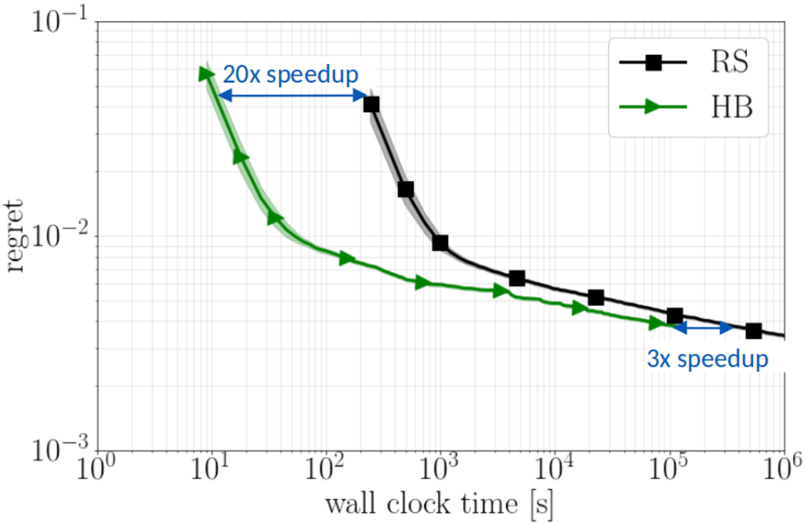


BOHB: Robust and Efficient HPO at Scale

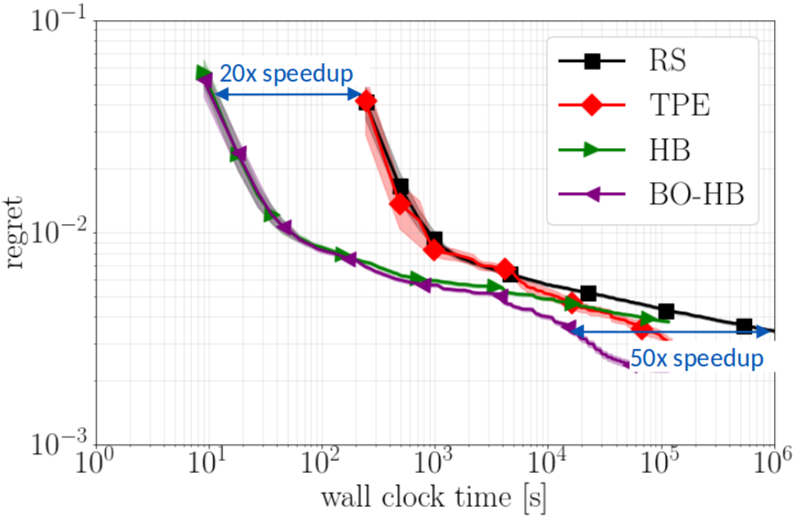
- Simple Combination of Bayesian Optimization and HyperBand
  [Falkner et al, ICML 2018]
  - Bayesian optimization for selecting configurations (a TPE-like variant)
  - Hyperband for selecting the budgets for them
- Advantages
  - Robust and efficient off-the-shelf tool
  - Strong anytime performance
  - Strong performance with larger budgets
  - Scalable to high dimensions, parallel workers, different parameter types (categorical, integer, continuous)
  - BSD-licensed, on Github: `https://github.com/automl/HpBandSter`
- Disclaimer: I'm a coauthor, so I am biased.

Random search vs. TPE




20x speedup
3x speedup
RS
HB
regret
wall clock time [s]
10^{-1}
10^{-2}
10^{-3}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}

BOHB achieves the best of both worlds



Questions to Answer for Yourself / Discuss with Friends

- Repetition: List three methods to speed up blackbox HPO
- Discussion: When will random search outperform successive halving?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

Hyperparameter Gradient Descent

Frank Hutter Abhinav Valada

University of Freiburg

HPO as a bilevel optimization problem

- Let $\mathcal{L}_{val}(\boldsymbol{w}, \boldsymbol{\theta})$ denote the validation loss of a network with weights $\boldsymbol{w}$ and hyperparameters $\boldsymbol{\theta}$; likewise, $\mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\theta})$ is the training loss.
- Then, the optimization of $\boldsymbol{\theta}$ can be written as the following bilevel optimization problem [Franceschi et al., 2018]:

$$\boxed{\begin{array}{c} \min_{\boldsymbol{\theta}} \mathcal{L}_{\text{val}}(\boldsymbol{w}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) \\ s.t.\ \boldsymbol{w}^*(\boldsymbol{\theta}) \in \operatorname{argmin}_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{w}, \boldsymbol{\theta}) \end{array}}$$

Gradients for Hyperparameters

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathsf{val}}(\boldsymbol{w}^*(\boldsymbol{\theta}), \boldsymbol{\theta})$$
$$s.t.\ \boldsymbol{w}^*(\boldsymbol{\theta}) \ \in \ \mathrm{argmin}_{\boldsymbol{w}} \mathcal{L}_{\mathsf{train}}(\boldsymbol{w}, \boldsymbol{\theta})$$

- We can compute gradients for $\boldsymbol{\theta}$ by differentiating through the entire SGD optimization run that leads to $\boldsymbol{w}^*(\boldsymbol{\theta})$ [MacLaurin et al, 2015]
- This yields a lot of freedom to include hyperparameters that would otherwise be really unwieldy
- Hot topic with lots of recent work [Pedregosa, 2016, Luketina et al, 2016, Francesci et al, 2017, Franceschi et al., 2018, Baydin et al, 2018, Mackay et al, 2019, Lorraine et al, 2020]

Approximation of the Bilevel Optimization Problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{val}}(\boldsymbol{w}^*(\boldsymbol{\theta}), \boldsymbol{\theta})$$
$$s.t.\ \boldsymbol{w}^*(\boldsymbol{\theta}) \in \operatorname{argmin}_{\boldsymbol{w}} \mathcal{L}_{\text{train}}(\boldsymbol{w}, \boldsymbol{\theta})$$

- Interleave optimization steps [Luketina et al, 2016]

Hyperparameter update step w.r.t. $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{val}}$
Parameter update step w.r.t. $\nabla_{\boldsymbol{w}} \mathcal{L}_{\text{train}}$

  - In general no guarantee of convergence, but often works surprisingly well

Questions to Answer for Yourself / Discuss with Friends

- Repetition: Write down the formulation of hyperparameter optimization as a bilevel optimization problem
- Repetition: In the alternating SGD approach by Luketina et al, 2016, which two steps are being alternated?

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 9: Hyperparameter Optimization

Summary, Further Reading, References

Frank Hutter Abhinav Valada

University of Freiburg

Summary by Learning Goals

Having heard this lecture, you can now ...

- Define the hyperparameter optimization problem
- Describe and apply manual ways of tuning DL hyperparameters
- Describe blackbox optimization methods for HPO
- Discuss various speedup techniques for HPO
- Explain the approach of multi-fidelity HPO methods
- Discuss the potential benefits of gradient-based HPO

Further Reading

Read the HPO survey [Feurer & Hutter, 2019], which is the main source for this week's material.

References

Canziani, A., Paszke, A. and Culurciello, E. (2016)
An Analysis of Deep Neural Network Models for Practical Applications
CoRR abs/1605.07678
https://arxiv.org/pdf/1605.07678.pdf

Goodfellow, I., Bengio, Y., Courville, A. (2016)
Deep Learning
MIT Press.
https://www.deeplearningbook.org/
Citation from Google Scholar

Wikimedia Commons (2006)
CMA-ES
https://en.wikipedia.org/wiki/CMA-ES#/media/File:Concept_of_directional_optimization_in_CMA-ES_algorithm.png

Brochu, Eric and Cora, Vlad M and De Freitas, Nando (2010)
A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning
arXiv preprint arXiv:1012.2599
https://arxiv.org/abs/1012.2599

Domhan, Tobias and Springenberg, Jost Tobias and Hutter, Frank (2015)
Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves
Twenty-fourth international joint conference on artificial intelligence
https://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/view/11468/11222

Klein, Aaron and Falkner, Stefan and Bartels, Simon and Hennig, Philipp and Hutter, Frank (2017)
Fast bayesian optimization of machine learning hyperparameters on large datasets
Artificial Intelligence and Statistics
http://proceedings.mlr.press/v54/klein17a/klein17a.pdf