

8th International Conference on Computers Communications and Control: ICCCC2020

Romania, Oradea, May 11-15, 2020

Practical Hyperparameter Optimization for Deep Learning

Răzvan Andonie¹

¹Computer Science Department, Central Washington University, USA

May 13, 2020

Overview

- 1 Hyperparameter Optimization in Deep Learning: The Problem
- 2 Methods for Hyperparameter Optimization
- 3 Search Space & Training Time Reduction
- 4 Weighted Random Search for Deep Learning Optimization
- 5 Conclusions and Open Questions

- 1 Hyperparameter Optimization in Deep Learning: The Problem
- 2 Methods for Hyperparameter Optimization
- 3 Search Space & Training Time Reduction
 - Instance selection
 - Hyperparameter selection/ranking by functional analysis of variance
 - Feature selection
 - Use additional objective functions
- 4 Weighted Random Search for Deep Learning Optimization
- 5 Conclusions and Open Questions

Definition (1)

Most Machine Learning models are described by two sets of parameters:

- 1 Regular parameters that are learned through training. For a neural network: the weights of the connections.
- 2 *Hyperparameters* which are set before the learning starts. For a neural network: number and type of layers, number and type of nodes, etc.

Definition (2)

- The model itself may be considered, at a meta-level, a hyperparameter. In this case, we have a list of possible models, each with its own list of hyperparameters, and all have to be optimized for a given problem.
- Some optimizers attempt to optimize both the choice of the model (model selection) and the hyperparameters of the model: Auto-WEKA, Hyperopt-sklearn, AutoML, auto-sklearn, etc.

The optimization problem - a byrd's eye view

The aim of *hyperparameter optimization* is to find the hyperparameters of a given model that return the best performance as measured on a validation set. This process can be represented in equation form as:

$$x^* = \underset{x \in \mathfrak{X}}{\operatorname{arg\,min}} f(x), \quad (1)$$

where $f(x)$ is an objective function to minimize (such as RMSE) evaluated on the validation set; x^* is the set of hyperparameters that yields the lowest value of the score, and x can take on any value in the domain \mathfrak{X} .

In simple terms, we want to find the model hyperparameters that yield the best score on the validation set metric.

The optimization problem - a byrd's eye view

The aim of *hyperparameter optimization* is to find the hyperparameters of a given model that return the best performance as measured on a validation set. This process can be represented in equation form as:

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{arg\,min}} f(x), \quad (1)$$

where $f(x)$ is an objective function to minimize (such as RMSE) evaluated on the validation set; x^* is the set of hyperparameters that yields the lowest value of the score, and x can take on any value in the domain \mathcal{X} .

In simple terms, we want to find the model hyperparameters that yield the best score on the validation set metric.

The detailed optimization problem

More detailed, eq. (1), can be written as:

$$x^* = \underset{x \in \mathbb{X}}{\operatorname{arg\,min}} f(x, \gamma^*; S_{\text{validation}}) \quad (2)$$

Eq. (2) includes an inner optimization used to find γ^* , the optimal value of γ , for the current x value:

$$\gamma^* = \underset{\gamma \in \Gamma}{\operatorname{arg\,min}} f(x, \gamma; S_{\text{train}}) \quad (3)$$

$S_{\text{validation}}$ and S_{train} denote the validation and training datasets respectively; γ is the set of learned parameters in the domain Γ through minimization of the training error.

Validation is complex, since we have to evaluate the expectation of the score over an unknown distribution.

- Each time we try different hyperparameters for the objective function, we train a model on the training data, make predictions on the validation data, and then calculate the validation metric.
- The training + evaluation of a model with one set of hyperparameter values is called a *trial*.
- Each trial is computationally expensive, since it involves re-training the model.

The problem

In *deep learning* hyperparameter optimization, we typically have:

- A high-dimensional search space, which according to the curse of dimensionality principle, has to be covered with an exponentially increasing number of points (combinations of hyperparameters).
- Very large training sets.

Our Problem: How can we practically reduce the computational complexity of hyperparameter optimization in deep learning?

The problem

In *deep learning* hyperparameter optimization, we typically have:

- A high-dimensional search space, which according to the curse of dimensionality principle, has to be covered with an exponential increasing number of points (combinations of hyperparameters).
- Very large training sets.

Our Problem: *How can we practically reduce the computational complexity of hyperparameter optimization in deep learning?*

Factors (1)

The performance of a parameter optimization method is determined by the following factors:

- F1. The execution time of each trial.
- F2. The number of evaluated combinations of hyperparameters (the number of trials).
- F3. The performance of the search procedure. Within the same number of trials, different optimization methods can achieve different scores, depending on how "smart" they are.

Factors (2)

Search space reduction (F2) and search strategy (F3) are inter-connected and can be addressed in a sequence:

- F2 is a quantitative criterion (how many). For instance, we can first reduce the number of hyperparameters (F2), and create this way more flexibility in the following stage for F3.
- F3 is a qualitative criterion (how "smart"). For instance (F3), we can first rank and weight the hyperparameters based on the functional analysis of the variance of the objective function, and then reduce the number of trials (F2) by giving more chances to the more promising trials.
- There is a trade-off between F2 and F3.

Factors (2)

Search space reduction (F2) and search strategy (F3) are inter-connected and can be addressed in a sequence:

- F2 is a quantitative criterion (how many). For instance, we can first reduce the number of hyperparameters (F2), and create this way more flexibility in the following stage for F3.
- F3 is a qualitative criterion (how "smart"). For instance (F3), we can first rank and weight the hyperparameters based on the functional analysis of the variance of the objective function, and then reduce the number of trials (F2) by giving more chances to the more promising trials.
- There is a trade-off between F2 and F3.

- 1 Hyperparameter Optimization in Deep Learning: The Problem
- 2 Methods for Hyperparameter Optimization**
- 3 Search Space & Training Time Reduction
 - Instance selection
 - Hyperparameter selection/ranking by functional analysis of variance
 - Feature selection
 - Use additional objective functions
- 4 Weighted Random Search for Deep Learning Optimization
- 5 Conclusions and Open Questions

Best-known hyperparameter optimization methods

- Manual tuning
- Grid search
- Random search
- Gradient-free optimization (Nelder-Mead, Simulated Annealing, Evolutionary Algorithms, Particle Swarm Optimization, etc)
- Bayesian optimization

Software for Hyperparameter Optimization

Several software libraries dedicated to hyperparameter optimization exist, some of them being autonomous, while others being built on top of existing ML software.

- LIBSVM and scikit-learn come with their own implementation of GS, with scikit-learn also offering support for RS
- BayesianOptimization, Spearmint, pyGPGO
- Hyperopt-sklearn, Optunity

Several automated machine learning tools (AutoML) were developed. Their goal is to automatize the choice of the optimal pipeline for a labeled input dataset: data preprocessing, feature selection/extraction, and learning model with its optimal hyperparameters.

- Auto-WEKA, Auto-sklearn
- MLBox, auto-sklearn, Tree-Based Pipeline Optimization Tool, H2O, AutoKeras, TransmogrifAI

Cloud-based services for AutoML

Commercial cloud-based AutoML services offer highly integrated hyperparameter optimization capabilities:

- Google Cloud AutoML¹
- Microsoft Azure ML²
- Amazon SageMaker Automatic Model Tuning³

¹<https://cloud.google.com/automl/>

²<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/tune-model-hyperparameters>

³<https://aws.amazon.com/sagemaker/>

- 1 Hyperparameter Optimization in Deep Learning: The Problem
- 2 Methods for Hyperparameter Optimization
- 3 Search Space & Training Time Reduction**
 - Instance selection
 - Hyperparameter selection/ranking by functional analysis of variance
 - Feature selection
 - Use additional objective functions
- 4 Weighted Random Search for Deep Learning Optimization
- 5 Conclusions and Open Questions

The factors influencing the computational complexity of the optimization (review)

- F1. The execution time of each trial.
- F2. The total number of trials to be executed if using exhaustive search (the size of the search space).
- F3. The performance of the search procedure.

Computational complexity issues

To address these issues and reduce the search space, some standard techniques may be used:

- *Instance selection*: reduce the training dataset based on statistical sampling (relates to F1).
- *Feature selection* (relates to F1).
- *Hyperparameter selection*: select the most important hyperparameters for the model optimization (relates to F2 & F3).
- *Hyperparameter ranking*: detect which hyperparameters are more important for the model optimization and weight them (relates to F3 & F2).
- *Use additional objective functions*: number of operations, optimization time, etc (relates to F3 & F2).

We will analyze how some of these techniques can be used.

Computational complexity issues

To address these issues and reduce the search space, some standard techniques may be used:

- *Instance selection*: reduce the training dataset based on statistical sampling (relates to F1).
- *Feature selection* (relates to F1).
- *Hyperparameter selection*: select the most important hyperparameters for the model optimization (relates to F2 & F3).
- *Hyperparameter ranking*: detect which hyperparameters are more important for the model optimization and weight them (relates to F3 & F2).
- *Use additional objective functions*: number of operations, optimization time, etc (relates to F3 & F2).


We will analyze how some of these techniques can be used.

Instance selection

- Instance selection aims to select subset of the training set, hoping that it can represent the whole training set and achieve acceptable performance.
- For hyperparameter optimization, since we evaluate different ML models, we are only interested in filter (not in wrapper) instance selection: the selection criterion is based on the training data itself, not on the model.

Active learning

- A related approach to instance selection is *active learning* which sequentially identifies critical samples to train on.
- Bengio *et al.*⁴ suggested that guiding a classifier by presenting training samples in an order of increasing difficulty can improve learning.
- These ordering strategies can be seen as a special form of transfer learning where the initial tasks are used to guide the learner so that it will perform better on the final task.

⁴Y. Bengio, J. Louradour, R. Collobert, and J. Weston, *Curriculum Learning*, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009. 

The training sequence order as a hyperparameter

- At the extreme, we may consider the order of the training sequence order as a hyperparameter of the model.
- However, finding the optimal permutation based on some performance criteria is computationally not feasible, since it is in the order of the possible permutations.

Reduce/rank the number of hyperparameters

- How important is each of the hyperparameters, and how do their values affect performance? Which hyperparameter interactions matter?
- Having even partial answers to these questions may allow us to rank the hyperparameters based on their importance. This can significantly reduce the computational complexity of the optimization.
- For instance, hyperparameters can be ranked (and selected) based on the functional analysis of the variance of the objective function (sensitivity analysis).

Reduce the number of features

- By reducing the number of features (*feature selection*) we reduce training time.
- We may consider feature selection as a sub-case of hyperparameter selection. The combined optimization of features + hyperparameters increases significantly the computational complexity of the search process.
- A sequential greedy like search in two stages (features selection followed by hyperparameter selection) is simpler than a combined search and may be acceptable.

Use additional objective functions

- Beside accuracy, we may use additional objective function. The search could be also guided with goals like training time or memory requirements of the network.
- One possibility is to add an objective function measuring the complexity of the model. Smaller complexity also means a smaller number of hyperparameters. We prefer models with small complexities.
- Models with lower complexity can be trained faster. They also require smaller training sets (the curse of dimensionality) and have a smaller chance to over-fit (over-fitting increases with the number of hyperparameters).



Use additional objective functions

- Beside accuracy, we may use additional objective function. The search could be also guided with goals like training time or memory requirements of the network.
- One possibility is to add an objective function measuring the complexity of the model. Smaller complexity also means a smaller number of hyperparameters. We prefer models with small complexities.
- Models with lower complexity can be trained faster. They also require smaller training sets (the curse of dimensionality) and have a smaller chance to over-fit (over-fitting increases with the number of hyperparameters).

- 1 Hyperparameter Optimization in Deep Learning: The Problem
- 2 Methods for Hyperparameter Optimization
- 3 Search Space & Training Time Reduction
 - Instance selection
 - Hyperparameter selection/ranking by functional analysis of variance
 - Feature selection
 - Use additional objective functions
- 4 **Weighted Random Search for Deep Learning Optimization**
- 5 Conclusions and Open Questions

Weighted Random Search (1)

- We introduced⁵ the *Weighted Random Search* (WRS) method, a combination of Random Search (RS) and probabilistic heuristic .
- Instead of a blind RS search, the WRS method uses information from previous trials to guide the search process toward next interesting trials.
- WRS finds the global optimum faster than RS: on average, WRS converges faster than RS. We also proved that WRS is convergent.

⁵Florea AC, Andonie R. *Weighted Random Search for Hyperparameter Optimization*, International Journal of Computers, Communications & Control, 14, 2019, 154–169.  

Weighted Random Search (2)

- Instead of always generating new values (like in RS), the WRS algorithm uses for a certain number of dimensions the so far best obtained values.
- The exact number of dimensions that actually change at each iteration is controlled by the probabilities of change assigned to each dimension.
- Focusing on factor F3, WRS attempts to have a good coverage of the variation of the objective function. It determines the importance (the weight) of each parameter by computing the variation of the objective function.

Weighted Random Search (3)

- WRS first generates some test values by evaluating f for a set of randomly chosen inputs.
- Then it uses the obtained data to run fANOVA and rank the variables (the hyperparameters) of f .
- It assigns a greater probability of change to the variables with greater weight. For a parameter which produces a small variation of F , the probability of change is also small.

Weighted Random Search (4)

- The WRS method outperformed several state-of-the-art hyperparameter optimization methods (RS, Nelder-Mead, Particle Swarm Optimization, Sobol Sequences, Bayesian Optimization, and Tree-structured Parzen Estimator) for Convolutional Neural Network (CNN) parameter optimization⁶.
- The criterion used was the classification accuracy achieved within the same number of tested combinations of parameter values.
- The code is publicly available on GitHub⁷.

⁶Andonie, R., Florea, A. *Weighted Random Search for CNN Hyperparameter Optimization*, International Journal of Computers, Communications & Control, 15, 2020, ISSN 1841-9844.

⁷<https://github.com/acflorea/goptim/tree/keras>


- 1 Hyperparameter Optimization in Deep Learning: The Problem
- 2 Methods for Hyperparameter Optimization
- 3 Search Space & Training Time Reduction
 - Instance selection
 - Hyperparameter selection/ranking by functional analysis of variance
 - Feature selection
 - Use additional objective functions
- 4 Weighted Random Search for Deep Learning Optimization
- 5 Conclusions and Open Questions

Conclusions and open questions (1)

- Determining the proper architecture design for deep learning models is a challenge because it differs for each dataset and therefore requires adjustments for each one.
- For most datasets only a few of the hyperparameters really matter. There is no mathematical method for determining the appropriate hyperparameters for a given dataset, so the selection relies on trial and error.
- Our thesis is that for each problem we have to use a combination of complexity reduction techniques.

Conclusions and open questions (2)

- Does this mean that in general we cannot optimize the hyperparameters of a given model?
- For some given problems, recent automatic methods have produced results exceeding those accomplished by human experts⁸.
- We should be optimistic, but we should be also aware of the scalability limits, since in the most general case, we are dealing with a computationally hard optimization problem.

⁸J. Bergstra *et al.*, *Algorithms for Hyper-parameter Optimization*, NIPS, 2011. 

Thank you for your attention!

