OPT2020: 12th Annual Workshop on Optimization for Machine Learning

# An approximate gradient based hyper-parameter optimization in a neural network architecture

**Lakshman Mahto** LM.OPTLEARNING@GMAIL.COM
**Arun Chauhan** ARUNTAKHUR@GMAIL.COM

**Indian Institute of Information Technology Dharwad**

## Abstract

As we know, most of the learning algorithms arising from scientific applications involve hyper-parameters which control model structure in order to minimize testing error (i.e. to achieve minimum loss or maximum accuracy of the objective network). The objective and constraint are available only as the output of a black-box or simulation oracle that does not provide derivative information. In this paper, to find the best architecture of a neural network architecture to classify cat and dog images, we purpose an approximate gradient based method for optimal hyper-parameters setting which is efficacious than both grid search and random search. Unlike grid search which is exhaustive, the proposed method only searches in its nearest neighbourhood thus reduces computational time. Comparing with random search which does not guarantee convergence, the proposed method converges. Several hyper-parameter settings using CNN architecture with reference to LeNet-5 and a MNIST datasets are performed for cat and dog classification. And the numerical results demonstrated that random search takes 5 hour to give 82% accuracy, but our method takes 2 hour to give the same accuracy.

## 1. Introduction

Based on nested hierarchical representation of concepts and depth of the architecture, deep learning performs representation learning and machine learning in together. With this ability, deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as computer vision, image recognition, speech recognition, social network filtering, natural language processing, and machine translation [5]. Most deep learning algorithms come with several hyper-parameters to control the model complexity, which makes tuning of hyper-parameters an intractable task. However, coming up with an ideal model with good overall performance involves choosing of various hyper-parameters e.g. architectures of the deep neural networks, activation functions and learning rates, momentum, number of iterations etc. At smaller search space, DNN hyper-parameters can be adjusted using manual search, grid search, or random search [2, 3, 7, 8], as search space expands exponentially relative to the number of hyper-parameters more sophisticated hyper-parameter optimization methods are required. For DNNs model, searching of an optimal set of hyper-parameter known as hyper-parameter optimization and the process is a black-box optimization (BBO) problem. Because hyper-parameters can not be estimated to minimize the same cost function as the model parameters, since this would favour models with excessive complexity. For example, if regularization parameters were chosen to minimize the same loss as model parameters, then models with no regularization would always yield the smallest loss. For this reason, hyper-parameter opti-

mization algorithms seek to optimize a criterion of model quality which is different from the cost function used to fit model parameters. This criterion can be a goodness of fit on unseen data, such as a *cross-validation* loss, or some criteria of model quality on the train set.

In this paper we append and apply an approximate gradient based search on hyper-parameter space and try to reduce the overall computational expense and the time consumed for searching optimal hyper-parameter settings. Our contributions can be summarized as follows:
*Purposing a novel algorithm to find a approximate local minima of non-smooth and non-continuous error function on hyper-parameter space.*

## 2. Hyper-parameter optimization and related work

Hyper-parameter optimization is a black box optimization problems where the error $f$ is a black box function (i.e. we do not have an analytical expression for $f$ nor do we know its derivatives). The function $f$ maps a hyper-parameter choice $x$ of D configurable hyper-parameters to the validation error of a machine learning algorithm with learned parameters $\theta$. Optimizing $f$ as follows gives a way to automatically search for optimal hyper-parameters [4].

$$\min_{x \in \mathbb{R}^D} f(x, \theta, Z_{val}) \text{ s.t. } \theta = \arg\min_{\theta} f(x, \theta; Z_{train}), \tag{1}$$

where $Z_{train}$ and $Z_{val}$ denote the training and validation datasets, respectively, $\theta$ is learned by minimizing the training error, and $x$ is in a bounded set. Solving the problem (1) is challenging because of the high complexity of the function. The deep neural network is highly non-linear and not smooth because of its non-linear operators (drop-out and rectified linear unit operation). Therefore, the optimization of the classic method using the derivative of the function is impossible to apply. The derivative-free optimization, called black box optimization, could be a solution that does not utilize derivative information in the classical sense to find optimal solutions.

The most widely known methods in this issue are grid search and random search. When the grid search for the machine learning model is utilized for the optimization, it is possible to find the best hyper-parameter by setting the appropriate range and interval; however, a massive amount of time is required to learn the model. In particular, as the number of hyper-parameters increases, the number of grid points to be tried increases exponentially, making it almost impossible to find optimized hyper-parameter. Therefore, when the number of hyper-parameters is large, the grid search method has huge computational complexity. Therefore, a random search could be a solution as it gives an improved result of hyper-parameter optimization [3]. Random search is one of the simplest ways to optimize DNN hyper-parameters. This method iteratively generates hyper-parameter settings and evaluates the objective function. Random search has excellent parallelism and can handle integer and categorical hyper-parameters naturally. Bergstra and Bengio demonstrated that random search outperforms a manual search by a human expert and grid search [2, 6].

## 3. Purposed method

### 3.1. An approximate gradient based method

Here, in this work, we purpose an approximate gradient based method to find a optimal hyper-parameters setting of a standard neural network architecture. Approximated gradient can be estimated on predefined set of values taken from the grid on hyper-parameter space using either finite

differences or linear interpolation or Gaussian smoothing and smoothing. The purposed method starts by setting a non-uniform grid, where each dimension is a different hyper-parameter and each coordinate of a grid is hyper-parameter value. After setting the grid, we randomly initialize a point in the grid and calculate the initial accuracy (say InitialAcc) of chosen neural network architecture on the randomly selected initial point. Next, we find neighbouring values of the initialised point and calculate the accuracy of the neural network architecture on these neighbouring points. Now, we select a neighbouring point with maximum accuracy (say MaxAcc) based on approximate gradient and compare with InitialAcc. If MaxAcc is less than InitialAcc, then hyper-parameter corresponding to InitialAcc is the optimal hyper-parameter value else Max acc is greater than InitialAcc then hyper-parameter corresponding to MaxAcc would be starting value for the next step. By repeating this, we assume that the algorithm converges and gets stuck in the vicinity of optimal value.

**Algorithm 1:** Algorithm for approximate gradient based HPO of a neural network architecture.
**Input :** Grid point from hyper-parameter space
**Output:** Optimal hyper-parameters setting

Initialization: Choose $x_0$ and step size $\tau_0$
**for** *($i = 1$ to m)* **do**
    Poll step based on approximated gradient descent
    update
**end**

## 4. Experiment set up

### 4.1. Convergence to an optimal hyper-parameter setting of a NN architecture

Here, we performed an empirical experiment of our purposed method on a neural network architecture in which the cost function is the validation error with a five dimensional non uniform grid. Each dimension of the non uniform grid represents a one hyper-parameter. We set up a five layer neural network architecture where first layer consist of 12888 inputs, second layer is a hidden layer with 20 hidden units and Relu activation function, third layer is with 7 hidden units and Relu activation function, fourth layer is with 5 hidden units and sigmoid as activation function and the final layer is a single output unit. Among the hyper-parameter, two are two continuous (learning rate and drop-out rate) and four are integer (no. of hidden units in 1st hidden layer, no. of hidden layers in 2nd hidden layer, choice of activation functions and number of iterations). Again we initialize randomly a point in the non-uniform grid and calculate validation error at the initial point. Now we calculate validation error at all the 10 neighbouring points of the initial point. The neighbouring point with minimum validation error is compared with the validation error at the initial point. If the minimum validation error of the neighbouring point is relatively less the validation error at the initial point, it replace the initial point for next next step similar to first first experiment. We keep on repeating above steps till the convergence of algorithm. After performing several such experiment several times, we get several minimum values of validation error. The point corresponding to the overall minimum validation error is our optimal hyper-parameter point. From this experiment, we conclude that the algorithm is convergence for non uniform grids as well.

### 4.2. Dataset used

Cat vs Non-cat dataset which is used by Andrew Ng for teaching Deep Learning course on coursera [9]. There are 209 training images in which X are cat images and Y are non-cat images. For testing there are 50 test images in which X are cat images and Y are non-cat images. The dimension of each images in both training and testing dataset are $64 \times 64 \times 3$ which results in flat vector of dimension $12,228 \times 1$.

### 4.3. Machine configuration

Processor : Intel(R) Core i7 $-$ 6700HQ CPU @$2.60GHz$, RAM : $4.00GB$, System Type : 64-bit operating system.

### 4.4. Hyper-parameters of the framework

A variety of hyper-parameters must be chosen to tune a DNN for a given application.These hyper-parameters affect different aspects of the network: the architecture, the optimization process and the handling of the data. The following section lists the hyper-parameters considered in this study along with their respective types and scopes.

### 4.5. The network architecture

A convolutional neural network (CNN) is a deep neural network consisting of a succession of convolution layers followed by fully connected layers. Table (4.5) summarizes the hyper-parameters responsible for defining the structure of the neural network along with performance comparison of various methods for hyper-parameter optimization.

| Hyperparameter | Range | Type |
|---|---|---|
| No of iterations (ITR) | [100,2000] | integer |
| Learning rate (LR) | [0.0034,0.0078] | contin. |
| # units in 1st hidden layer,$h_1$ | [1.30] | integer |
| # units in 1st hidden layer,$h_2$ | [1,10] | integer |
| # units in 1st hidden layer,$h_3$ | [1,6] | integer |

| Method | Time | Acc |
|---|---|---|
| Grid search (4) | 10 | 84 |
| Rand. search(4) | 5 | 82 |
| Our method (4) | 2 | 82 |
| Grid search (5) | 12 | 82 |
| Rand. search(5) | 5 | 80 |
| Our method (5) | 2 | 80 |

Table 1: Various hyper-parameter for the neural network structure and performance comparison of various methods for hyper-parameter optimization

## 5. Result

Firstly, we performed hyper-parameters tuning on our neural network architecture by grid search. with cat vs non-cat dataset where training images were 209 and validation images were 50. The chosen hyper-parameters were learning rate, number of Iteration, number of units in the first hidden layer, number of units in the second hidden layer and number of units in the third hidden layer. Five different value of each hyper-parameter were taken in each dimension. As every dimension

had 5 different values and there are 5 dimensions, a total of 3125 different combinations of hyper-parameters were there. The search iterated over 3125 times. And the grid search took approximately 12 hours with the best validation accuracy 84%.

Secondly, we performed hyper-parameter tuning on the same architecture based on random search RandomSearchCV of scikit-learn [1] on the same cat vs non-cat dataset as above with best validation accuracy of 82% in approximately 5 hours.

Again we perform hyper-parameter tuning on same architecture and on the same dataset as above using our purposed method, with best validation accuracy of 82% in approximately 2 hours. Hyper-parameter optimization results on 4- dimension and 5-dimension are summarized below in the following tables:

| GP No | ITR | LR | $h_1$ | $h_2$ | Acc |
|---|---|---|---|---|---|
| 1 | 100 | 0.0078 | 15 | 3 | 34 |
| 2 | 100 | 0.0077 | 15 | 9 | 34 |
| 3 | 300 | 0.0045 | 25 | 9 | 60 |
| 4 | 300 | 0.0077 | 20 | 7 | 78 |
| 5 | 400 | 0.0077 | 25 | 3 | 44 |
| 6 | 300 | 0.0077 | 30 | 5 | 34 |
| 7 | 700 | 0.0077 | 20 | 9 | 80 |
| 8 | 300 | 0.0034 | 20 | 9 | 34 |
| 9 | 400 | 0.0077 | 10 | 5 | 34 |
| 10 | 400 | 0.0077 | 15 | 3 | 34 |
| 11 | 400 | 0.0045 | 20 | 5 | 66 |
| 12 | 400 | 0.0077 | 15 | 3 | 34 |
| 13 | 300 | 0.0077 | 15 | 10 | 46 |
| 14 | 700 | 0.0077 | 30 | 5 | 56 |
| 15 | 700 | 0.0034 | 20 | 7 | 78 |
| 16 | 400 | 0.0077 | 10 | 10 | 84 |

| GP No | ITR | LR | $h_1$ | $h_2$ | $h_3$ | Acc |
|---|---|---|---|---|---|---|
| 1 | 300 | 0.0077 | 15 | 7 | 6 | 78 |
| 2 | 500 | 0.0077 | 15 | 7 | 6 | 76 |
| 3 | 100 | 0.0077 | 15 | 7 | 6 | 62 |
| 4 | 300 | 0.0034 | 15 | 7 | 6 | 66 |
| 5 | 300 | 0.0077 | 25 | 7 | 6 | 36 |
| 6 | 300 | 0.0077 | 15 | 9 | 6 | 58 |
| 7 | 300 | 0.0077 | 30 | 7 | 6 | 42 |
| 8 | 300 | 0.0077 | 15 | 5 | 6 | 82 |
| 9 | 300 | 0.0077 | 15 | 7 | 5 | 42 |
| 10 | 500 | 0.0077 | 15 | 5 | 6 | 76 |
| 11 | 100 | 0.0077 | 15 | 5 | 6 | 74 |
| 12 | 300 | 0.0034 | 15 | 5 | 6 | 68 |
| 13 | 300 | 0.0077 | 25 | 5 | 6 | 72 |
| 14 | 300 | 0.0077 | 30 | 5 | 6 | 40 |
| 15 | 300 | 0.0077 | 15 | 3 | 6 | 36 |
| 16 | 300 | 0.0077 | 15 | 5 | 5 | 70 |

Table 2: Results of Grid search with four hyper-parameters and approximate gradient method with five hyper-parameters respectively for hyper-parameter optimization of a neural network architecture.

## 6. Conclusion

From the above results we conclude that grid search gave best result of 84% test accuracy in 12 hours, which is too long. Random searches gave 82% test accuracy which is slightly less than optimal value but it only took 5 hours and our method gave approximately 82% test accuracy in only 2 hours which is very less time. So we conclude that our method gives best results in very less time on optimal hyper-parameter value.

| GP No | ITR | LR | $h_1$ | $h_2$ | $h_3$ | Acc |
|---|---|---|---|---|---|---|
| 1 | 700 | 0.0078 | 20 | 3 | 4 | 52 |
| 2 | 500 | 0.0045 | 30 | 9 | 4 | 78 |
| 3 | 100 | 0.0045 | 15 | 3 | 2 | 38 |
| 4 | 700 | 0.0075 | 25 | 3 | 4 | 70 |
| 5 | 100 | 0.0045 | 20 | 7 | 2 | 68 |
| 6 | 700 | 0.0078 | 20 | 5 | 4 | 80 |
| 7 | 400 | 0.0045 | 15 | 3 | 5 | 34 |
| 8 | 400 | 0.0045 | 15 | 9 | 2 | 40 |

| GP No | ITR | LR | $h_1$ | $h_2$ | $h_3$ | Acc |
|---|---|---|---|---|---|---|
| 1 | 400 | 0.0045 | 15 | 3 | 5 | 34 |
| 2 | 700 | 0.0078 | 20 | 10 | 5 | 80 |
| 3 | 500 | 0.0078 | 25 | 3 | 2 | 34 |
| 4 | 700 | 0.0078 | 15 | 3 | 6 | 76 |
| 5 | 100 | 0.0075 | 20 | 10 | 5 | 34 |
| 6 | 700 | 0.0077 | 20 | 5 | 5 | 50 |
| 7 | 500 | 0.0034 | 25 | 3 | 4 | 34 |
| 8 | 700 | 0.0045 | 20 | 7 | 3 | 64 |
| 9 | 300 | 0.0034 | 10 | 10 | 1 | 34 |
| 10 | 400 | 0.0034 | 25 | 5 | 2 | 42 |

Table 3: Results of random search with $1000$ and $2000$ iterations respectively for hyper-parameter optimization of a neural network with five hyper-parameters.

## References


[1] Bergstra, James, et al. "Theano: a CPU and GPU math expression compiler." Proceedings of the Python for scientific computing conference (SciPy). Vol. 4. No. 3. 2010.

[2] Bergstra, James S., et al. "Algorithms for hyper-parameter optimization." Advances in neural information processing systems. 2011.

[3] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of Machine Learning Research 13.Feb (2012): 281-305.

[4] Ilievski, Ilija, et al. "Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates." Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[5] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[6] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. vol. 1, no. 2. Cambridge: MIT press, 2016.

[7] Swersky, Kevin, Jasper Snoek, and Ryan Prescott Adams. "Freeze-thaw Bayesian optimization." arXiv preprint arXiv:1406.3896 (2014).

[8] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems. 2012

[9] Andrew Ng course on deep learning in coursera, https://www.coursera.org/specializations/deep-learning.