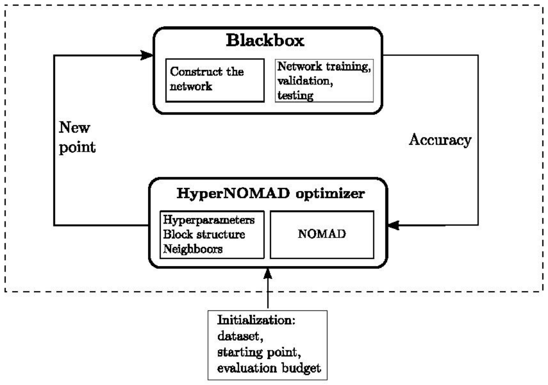
HYPERNOMAD Documentation

Release stable

Mar 06, 2020

Installation Guide

HyperNOMAD is a Python package dedicated to the hyperparameter optimization of deep neural networks. The package contains a blackbox specifically designed for this problematic and provides a link with the NOMAD software used for the optimization. The blackbox takes as inputs a list of hyperparameters, builds a corresponding deep neural network in order to train, validate and test it on a specific data set before returning the test error as a mesure of performance. NOMAD is then used to minimize this error. The following appendix provides an overview of how to use the HyperNOMAD package.


Blackbox
Construct the network
Network training, validation, testing
New point
Accuracy
HyperNOMAD optimizer
Hyperparameters Block structure Neighboors
NOMAD
Initialization: dataset, starting point, evaluation budget

CHAPTER 1

Prerequisites

In order to run HyperNOMAD correctly, please make sure to have:

- Python > 3.6
- PyTorch.
- GCC > 3.8
- A compiled version of NOMAD.

1.1 Check that the requirements are fullfiled

Here are simple tests to check that everything is set correctly before installing HyperNOMAD.

1.1.1 Pytorch

In order to test that Pytorch is correctly installed try the following command

```
python
>>> import torch
```

1.1.2 NOMAD

Try running the following command

```
nomad -info
```

For more help on the installation of NOMAD, please refer to the user_guide.

CHAPTER 2

Step by step installation of HyperNOMAD

The installation of HyperNOMAD can start once the package is downloaded and the prerequisites installed. The package contains a Makefile responsible for builinding the binaries. To start the installation, you need to execute the following command:

```
make
    building HyperNOMAD ...

    To be able to run the example
    the HYPERNOMAD_HOME environment variable
    must be set to the HyperNOMAD home directory
```

When the compilation is successful, a message appears asking to set an environment variable 'HYPERNOMAD_HOME'. This can be done by adding a line in the file .profile or .bashrc :

```
export HYPERNOMAD_HOME=hypernomad_directory
```

2.1 Check that the installation is successful

The executable hypernomad.exe is located in the bin directory. You can check that the installation is successful by trying to run the commad

```
$HYPERNOMAD_HOME/bin/./hypernomad.exe -i
```

which should return the following informations:

```
--------------------------------------------------
  HyperNomad - version 1.0
--------------------------------------------------
  Using Nomad version 3.9.0 - www.gerad.ca/nomad
--------------------------------------------------

```

(continues on next page)

(continued from previous page)

```
Run           : hypernomad.exe parameters_file
Info          : hypernomad.exe -i
Help          : hypernomad.exe -h
Version       : hypernomad.exe -v
Usage         : hypernomad.exe -u
Neighboors    : hypernomad.exe -n parameters_file
```

CHAPTER 3

Basic usage

HyperNOMAD is a library that aims at optimizing the hyperparameters of a deep neural network for a given application. Note that, at this stage, HyperNOMAD is tailored for convolutional networks only.

In order to start an optimization, a few informations must be provided in order to specify the dataset used and optionnally some informations on the search space. This is done in a parameter file which is passed as an argument to hypernomad.exe, as is shown in the following example:

```
$HYPERNOMAD_HOME/bin/./hypernomad.exe parameter_file.txt
```

3.1 Choosing a dataset

The parameter file must contain some mandatory informations such as the name of the data set and the number of times the blackbox is called, which corresponds to the number of different configurations HyperNOMAD is allowed to try. This package comes with the data sets that are listed in the table below.

Dataset	Training data	validation data	test data	Number of classes
MNIST	40000	10000	10000	10
Fashion MNIST	40000	10000	10000	10
EMNIST	40000	10000	10000	10
KMNIST	40000	10000	10000	10
CIFAR10	40000	10000	10000	10
CIFAR100	40000	10000	10000	100
STL10	4000	1000	8000	10

HyperNOMAD also offers the possibily of using a personnal data set in which case the user is responsible for providing the necessary informations to the blackbox. The necessary instructions to do so are provided in the Advanced usage section.

3.2 Specifying the search space

The user can choose to provide additionnal informations on the search space considered. HyperNOMAD allows for a good flexibility of tuning a convolutional network by considering multiple aspects of a network at once such as the architecture, the dropout rate, the choice of the optimizer and the hyperparameters related to the optimization aspect (learning rate, weight decay, momentum, . . .), the batch size, etc. The user can choose to optimize all these aspects or select a few and fixe the others to certain values. The user can also change the default range of each hyperparameter.

This information is passed through the parameter file by using a specific synthax:

```
KEYWORD INITIAL_VALUE LOWER_BOUND UPPER_BOUND FIXED/VAR
```

This table lists all the possible keywords, their interpretation and the default values and ranges for each one

Name	Description	Default	Range
DATASET	name of the dataset used	no default	From previous table, or CUSTOM
NUMBER_OF_CLASSES	number of classes of the problem	no default	Only use if dataset is CUSTOM
MAX_BB_EVALS	number of configurations to try	no default	Integer > 1
NUM_CON_LAYERS	number of convolutional layers	2	[0, 100]
OUTPUT_CHANNELS	number of output channels of the layer	6	[1, 100]
KERNELS	size of the kernels applied	5	[1, 20]
STRIDES	step of the kernels	1	[1, 3]
PADDINGS	size of the padding	0	[0, 2]
DO_POOL	weather apply a pooling or not	0	0, 1
NUM_FC_LAYERS	number of fully connected layers	2	[0, 500]
SIZE_FC_LAYER	size of the fully connected layer	128	[1, 1000]
BATCH_SIZE	the batch size	128	[1, 400]
OPTIMIZER_CHOICE	from SGD, Adam, Adagrad, RMSProp	3	[1, 4]
OPT_PARAM_1	learning rate	0.1	[0, 1]
OPT_PARAM_2	second parameter of the optimizer	0.9	[0, 1]
OPT_PARAM_3	third parameter of the optimizer	0.005	[0, 1]
OPT_PARAM_4	fourth parameter of the optimizer	0	[0, 1]
DROPOUT_RATE	probability of dropping a node	0.5	[0, 0.95]
ACTIVATION_FUNCTION	choice from ReLU, Sigmoid or Tanh	1	[1, 3]
REMAINING_HYPERPARAMETERS	use to fixe or vary those not listed in the parameter file	VAR	FIXED, VAR

3.3 Example of a parameter file

Here is an example of an acceptable parameter file. First, the dataset MNIST is choosen and we specify that HyperNOMAD is allowed to try a maximum of 100 configurations. Then, the number of convolutional layers is fixed throught the optimization to 5, the two '-' appearing after the '5' mean that the default lower and upper bounds are not changed. The kernels, number of fully connected layers and activation function are respectively initialized at 3, 6, and 2 (Sigmoid) and the dropout rate is initialized at 0.6 with a new lower bound of 0.3 and upper bound of 0.8 Finally, all the remaining hyperparameters from Table~ref{tab:keywords} that are not explicitly mentioned in this file are fixed to their default values during the optimization.

```
# Mandatory information
DATASET                  MNIST
MAX_BB_EVAL              100

# Optional information
NUM_CON_LAYERS           5  -  -  FIXED
KERNELS                  3
NUM_FC_LAYERS            6
ACTIVATION_FUNCTION      2
DROPOUT_RATE             0.6  0.3 0.8
REMAINING_HPS            FIXED
```

This parameter file is provided in the directory ‘examples’ from where we can execute the following command in order to run HyperNOMAD on this search space

```
$HYPERNOMAD_HOME/bin/./hypernomad.exe $HYPERNOMAD_HOME/examples/mnist_first_example.
↪txt
```

CHAPTER 4

Using X0 as a starting point

The main advantage of choosing this method of initialization rather than the previous one, which relies on using keywords, is that defining an X0 allows for more flexiblity since one can choose a value for each parameter of each layer. For example, using a keyword such as ‘KERNELS’ means that all the kernels applied on every convolutional layer will have the same initial value. Whereas an X0 allows to initialize each kernel individually.

The order and meaning of the variables in X0 is hardcoded in HyperNOMAD. Let’s use the following parameter file as an example :

```
DATASET MNIST
MAX_BB_EVAL 100

HYPER_DISPLAY 3

#              [ CONVOLUTION BLOCK               ]    [ FULLY CONNECTED BLOCK ] ␣
↪[BATCH] [   OPTIMIZER BLOCK      ] [DROPOUT][ACTIVATION]
X0           (   2     6   5 1 0 1    16   5 1 0 1        2   128    84               128␣
↪     3    0.1   0.9   0.0005 0      0.2         1        )
#LOWER_BOUND (   1     1   1 1 0 0     1   1 1 0 0        0     1     1                 1 ␣
↪      1    0     0     0     0       0         1        )
#UPPER_BOUND ( 100  1000  20 3 2 1  1000  20 3 2 1      500  1000  1000              400 ␣
↪      4    1     1     1     1       1         3        )


DROPOUT_RATE 0.5 - - FIXED
KERNELS 10 - - FIXED
REMAINING_HYPERPARAMETERS VAR
```

4.1 Analysis of the example

First, ‘HYPER_DISPLAY’ allows set the level of details on the steps of HyperNOMAD. The default value is 1, and the maximum is 3. Then, X0 is presented as a list of parameters that are respectively categorised into the convolutional block, the fully connected block, the batch size, the optimizer block, the dropout rate and the activate function.

The blocks for the batch size, dropout rate and activate function contain each one single value which that of the corresponding hyperparameter.

The first variable of the convolutional block indicates the number of convolutional layers : 2 in this example. Each convolutional layer has 5 associated variables : (number of output channeles, kernel, stride, padding, do pooling). Therefor, the first convolutional layer has 6 output channels, a (5,5) kernel, a stride of 1, no padding and performs a pooling afterwards. The same goes for the second layer.

The first variable of the fully connected block corresponds to the number of fully connected layers. The following variables indicate the size of each fully connected layer.

The first variable of the optimizer block indicates which optimizer is used, here is it Adagrad. The optimizer block always has 4 associated variables whose meaning change according to the optimizer chosen. For example in the case of SGD, the first variable is the learning rate followed by the momentum, the dampening and the weight decay.

4.2 Advantage of using X0

In addition to being able to initialize each hyperparameter on it's own, we can also define specific lower and upper bounds for each single hyperparameter as is shown in the previous example.

Note: Note that X0 takes precedence over the other keywords, therefore the tags KERNELS and DROPOUT_RATE will not affect this initial starting point.

CHAPTER 5

Using a personnal dataset

In addition to the datasets embedded in HyperNOMAD, the user can choose to use a personnal dataset by specifying the following informations in the parameter file:

```
DATASET CUSTOM
NUMBER_OF_CLASSES 20
```

When using a CUSTOM dataset, it is mandatory to provide HyperNOMAD with the number of classes. The user is also responsible of plugging the 3 datasets (training, validation and testing) into the blackbox. In the file blackbox.py, the lines 80 to 84 must be completed with the adequate information.

```
# Load the data
print('> Preparing the data..')

if dataset is not 'CUSTOM':
    dataloader = DataHandler(dataset, batch_size)
    image_size, number_classes = dataloader.get_info_data
    trainloader, validloader, testloader = dataloader.get_loaders()
else:
    # Add here the adequate information
    image_size = None
    number_classes = None
    trainloader = None
    validloader = None
    testloader = None
```

The image size is a tuple of the form : (number_input_channels, length_image, width_image). In the case of MNIST, the image size is (1, 28, 28).

The trainload, validloader and testloader must be instances of 'torch.utils.data.dataloader.DataLoader'.