OmniOpt

Norman Koch

August 6, 2021

Contents

1 You probably don't need this document!

Most probably, you do NOT need this document. For almost all purposes, using the GUI should be sufficient. You can find the GUI under `https://imageseg.scads.de/omnioptgui/`.

2 Author

The author of this wrapper and documentation is Norman Koch. If any problems occur, feel free to contact him via `norman.koch@tu-dresden.de`.

3 Goal

This python3-script can automatically minimize a given program with a arbitrary number of hyperparameters. Doing this, it optimizes it's own search space, i. e. it tries some numbers and then approximates better and better to the minimum.

It is based on the module `HyperOpt`[1].

4 Folders and files

4.1 Folders

4.1.1 documentation

Contains this document in LaTeX-format and as PDF.

4.1.2 projects

Contains the config files, programs, database-files and so on for all projects in several subfolders.

4.1.3 script

Contains the main-script and all of it's python modules.

4.1.4 test

Contains the test-suite for running automated tests.

4.1.5 projects

Contains the projects for running automated tests.

[1] `http://hyperopt.github.io/hyperopt/`

4.2 Files

4.2.1 ./tools/debug.sh

Contains features for dealing with several `bash`-problems, locally or on Taurus (like loading modules, checking for return values, printing output to the command line, ...).

4.2.2 ./script/keras_output_parser.pl

Creates a perl-representation of different learning stages inside a neural network created with Keras. (Still beta!)

4.2.3 ./sbatch.pl

The main script that launches the optimizer. Can be called via

4.2.4 ./tools/writeip.pl

Writes the IP of the main server, on which the MongoDB-client is started, into a log file, so workers on other nodes can connect to that one specific server.

4.2.5 ./script/getOpts.py

Offers functions for parsing and checking parameters. *Not to be called directly.*

4.2.6 ./script/mydebug.py

Contains functions for printing output and rudimentarily parses the command-line-options and the config-file-options, so that it can be determined whether to output debug-texts or not. *Not to be called directly.*

4.2.7 ./script/mypath.py

Central file for getting path-information (like the working path). *Not to be called directly.*

4.2.8 ./script/plot3.py

Plots a project's data graphically. Parameters:

```
--project=abc   Names the project that should run
--int=1         1 = Convert all values to integers,
                0 (or omitted): don't convert to integers
```

4.2.9 ./script/termcolor.py

For outputting colors in the terminal. *Not to be called directly.*

4.2.10 ./script/dbtocsv.py

Print's a project's database to a CSV-file or to the screen.

```
--project=abc           Names the project that should run
--filename=abc.csv      Writes it to the file abc.csv (optional)
```

4.2.11 ./script/hyperopt-mongo-worker

The worker script. *Not to be called directly.*

4.2.12 ./script/mongo_db_objective.py

The file that gets called from the worker. *Not to be called directly.*

4.2.13 ./script/myfunctions.py

Contains all kind of functions needed for the program's functionality. *Not to be called directly.*

4.2.14 ./script/myregexps.py

A central place for storing regular expressions often needed. *Not to be called directly.*

4.2.15 ./script/range_generator.py

Contains all kinds of ranges that can be specified through config.ini. *Not to be called directly.*

5 Parameters

The parameters are specified in the config.ini of the specific project, which is read by the main-script and the hyperopt-worker-script. The config.ini is splitted into several categories.

5.1 Data types

Every parameter line has a data type, which is in the curved brackets after the example line in the headline of every subsubsection. For example,

" `precision = 5 (integer)` " means the line in the config file must be like " `precision =` n ", where n is an integer (i. e. $n \in \mathbb{N}$).

Boolean types are not `true` and `false`, but 0 (= `false`) and 1 (= `true`). The datatype must not be put into the real config file!

5.2 Comments

In the config-file, everything after '#' is commented out, so that

```
[DIMENSIONS]
dimensions = 1

dim_0_name = cpuparam
range_generator_0 = hp.choice
options_0 = 0,1,2,3
#options_0 = 0,1,2,3 # this line will be ignored
```

5.3 [DATA]

5.3.1 precision = 5 (integer)

This sets the number of shown digits after the decimal point (but it does *not* make calculations more accurate).

5.3.2 max_evals = 60 (integer)

This sets the number of evaluations that should be tried out before ending the script.

5.3.3 objective_program = perl /test.pl ($x_0) int($x_1) (string)

Either this or objective needs to be set!

This sets the program that will run with $x_1 being replaced by the x-value of the zeroeth dimensions, $x_1 by the oneth dimension and so on. The output of the script should only be the result and nothing else, though you can e. g. grep and sed for a line containing only the resulting number and use that as objective_program, too. The script's name must be a fully qualifying path, not a relative one, for `hyperopt-mongo-worker` to find it.

Also, when writing `int($x_1)`, the value will be rounded down the the nearest integer.

The output of the program that runs must either only output it's value (that which should be minimized), or, alternatively, the output-line created by the script should look like this:

`RESULT: 3.14159265`

See Parameter substitution for details.

5.3.4 algo_name = tpe.suggest (string)

This sets the name of the algorithm that searches in the specified space.

The following algorithms can be used here:

tpe.suggest Tree of Parzen Estimators

hyperopt.rand.suggest Random Search

5.4 [DIMENSIONS]

5.4.1 dimensions = 2 (integer)

This sets the number of total dimensions beginning with dimension 0 the search goes through.

5.4.2 min_dim_0 = 0 and max_dim_0 = 20 (whole number)

Specifies the min. and max. values for each dimension. Cannot be left empty and must be integer. You can imagine the x_0-values as x-axis, x_1 and y and so on.

5.5 [RANGE_GENERATOR]

5.5.1 name = hp.uniform (string)

Sets the algorithm for the range-generator. The following algorithms are allowed:

hp.choice(label, options) Returns one of the options, which should be a list or tuple. The elements of options can themselves be [nested] stochastic expressions. In this case, the stochastic choices that only appear in some of the options become conditional parameters.

hp.pchoice(label, p_options) One of the option terms listed in p_options, a list of pairs (prob, option) in which the sum of all prob elements should sum to 1. The pchoice lets a user bias random search to choose some options more often than others.

hp.uniform(label, low, high) Uniformly between low and high. When optimizing, this variable is constrained to a two-sided interval.

hp.quniform(label, low, high, q) Drawn by round(uniform(low, high) / q) * q, Suitable for a discrete value with respect to which the objective is still somewhat smooth.

hp.loguniform(label, low, high) Drawn by exp(uniform(low, high)). When optimizing, this variable is constrained to the interval [e^{low}, e^{high}].

hp.qloguniform(label, low, high, q) By round(exp(uniform(low, high)) / q) * q. Suitable for a discrete variable with respect to which the objective is smooth and gets smoother with the increasing size of the value.

hp.normal(label, mu, sigma) A normally-distributed real value. When optimizing, this is an unconstrained variable.

hp.qnormal(label, mu, sigma, q) Drawn by round(normal(mu, sigma) / q) * q. Suitable for a discrete variable that probably takes a value around mu, but is technically unbounded.

hp.lognormal(label, mu, sigma) Drawn by exp(normal(mu, sigma)). When optimizing, this variable is constrained to be positive.

hp.qlognormal(label, mu, sigma, q) Drawn by round(exp(normal(mu, sigma))/q)*q. Suitable for a discrete variable with respect to which the objective is smooth and gets smoother with the size of the variable, which is non-negative.

hp.randint(label, upper) Returns a random integer in the range [0, upper). In contrast to quniform optimization algorithms should assume no additional correlation in the loss function between nearby integer values, as compared with more distant integer values (e. g. random seeds).

If other parameters are used, e. g. q, then you can set `q = 5` for example in the config-file.

5.6 [DEBUG]

5.6.1 debug = 1 (boolean)

Enables debugging-output. Set to 0 to disable.

5.6.2 debug_xtreme = 1 (boolean)

Enables even more debugging-output whereever possible. Set to 0 to disable.

5.6.3 info = 1 (boolean)

Enables informational-output whereever possible. Set to 0 to disable.

5.6.4 warning = 1 (boolean)

Enables warning-output whereever possible. Set to 0 to disable.

5.6.5 success = 1 (boolean)

Enables success-output whereever possible. Set to 0 to disable.

5.6.6 stack = 0 (boolean)

Shows the stack for every debug/info/warning message whereever possible, very useful for debugging. Set to 1 to enable.

5.7 [MONGODB]

5.7.1 worker_last_job_timeout = 1 (integer)

Sets the timeout for the hyperopt-mongo-worker.

5.7.2 poll_interval = 0.1 (float)

Sets the poll-interval for the hyperopt-mongo-worker.

5.7.3 kill_after_n_no_results = 10 (integer)

Sets up a process-wrapper for hyperopt-mongo-worker started as a fork such that after n "no job found"-messages the process gets killed.

6 MongoDB

MongoDB is used for storing results of done calculations. This may be used as a cache when re-doing similiar calculations (that is, calculcations having the same `dbname`-value), or to coordinate multiple machines to caculcate values and find the minimums.

7 Parameter substitution

In the `objective_program` section of the config-file, variables can be used. These are:

`($x_0)`, `($x_1)`, ..., `($x_`n`)`, which all get replaced with the dimension-values generated by `fmin`, and `$mainpath`, which get's replaced to the main-path of the script, that could, for example, be `/home/scads/omniopt/script`.

Also, you can use `$homepath` (which get's the path of the user's home directory) and `$projectpath`, which is the path of the project folder of the running instance.

7.1 Connect to MongoDB

Using the IP and port from the `config.ini`, you can connect to the database with

```
mongo --host 127.0.0.1:1234
```

(of course, depending on the server that should be used).
The following commands might be useful:

```
show dbs;
use myprojectname;
show collections;
db.jobs.find();
db.jobs.find().pretty();
db.jobs.find().limit(1);
db.jobs.find().pretty().limit(1);
db.jobs.find(
    {},
    { "result": 1, "misc.vals": 1 }
).sort( { "result.loss": 1 } ).pretty();
```

8 Get information from inside the script

Whenever you have MongoDB enabled, you run a program and it has lines like

```
q: 402
```

in it's output, you can get those data via the `bash dostuff.sh`. All data in this form (*any number of letters: number*) gets outputted as CSV-file into STDOUT. Just `grep` for `>>>OUTPUTHEADER>>>` and `>>>OUTPUT>>>`. The OUTPUTHEADER contains the names of the columns and the OUTPUT contains the columns itself.

You may have to remove multiple OUTPUTHEADER lines from the Output.

9 Automated testing

OmniOpt offers a near-complete test-suite that can be run automatically by starting `perl sbatch.pl -run_full_tests -debug`.

10 Get results

You can get all the results via the `bash dostuff.sh` without tinkering or memorizing commands.

11 Troubleshooting

11.1 EXCEPTION <class 'AttributeError'> 'NoneType' object has no attribute 'uniform'

If this occurs, you might created a new job and used the same name/mongo-db-data from an old one which is incompatible. Consider moving or deleting all the mongo-db-folders (e. g. `projects/projectname/mongodb`) or create a new jobname.

11.2 Cannot find `hyperopt-mongo-worker`

Check if the shebang-line is correct and points to a valid python3 program. E. g.

```
#!/usr/bin/env python3
```

Also make sure that the script is executable. This can be done with

```
chmod +x script/hyperopt-mongo-worker
```