benchopt

Benchmarking optimization algorithms

Ghislain DURIF (adapted from J. Salmon slides)

January 22nd, 2021

IMAG – CNRS

Credits

A. Gramfort

Inria Parietal

J. Salmon

Univ. Montpellier

M. Massias

Univ. of Genova

T. Dupré la Tour

UC Berkeley

T. Moreau (the boss)

Inria Parietal

Benchmarking algorithms in practice

Purpose: choosing the best algorithm to solve an optimization problem (especially related to a statistical or machine learning application)

Issues: numerous conditions and configurations including

- The **properties**, **scale**, **conditionning** and **generation process** of the data
- The **parameters** and **regularisation** of the objective
- The **efficiency**, **complexity** and **language** of the implementation

Solution: an impartial selection or comparison requires **a time consuming benchmark**!

benchopt

- Complete and versatile tool to quickly and efficiently design benchmarks to impartially and fairly compare algorithms solving an optimization problem

- Useful for **software design**, **publication writing**, **publication review**, etc.

See `https://benchopt.github.io/`

Using `benchopt` to run a benchmark

- Example benchmark for the ℓ_2 regularized logistic regression problem with multiple solvers and datasets

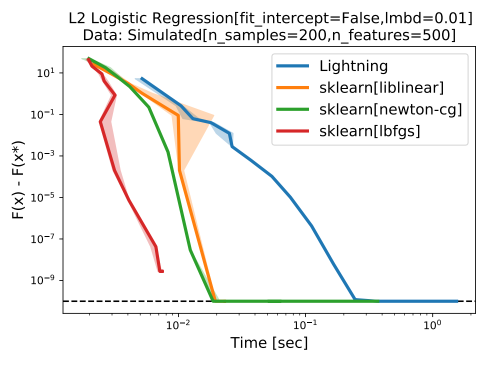
Commands to get the default benchmark and run it:

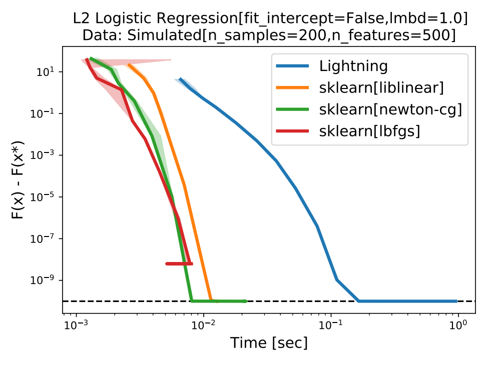
```
git clone https://github.com/benchopt/benchmark_logreg_l2
benchopt run ./benchmark_logreg_l2
```

- `benchopt` runs (with repetitions) each selected solver on each selected dataset with each selected parameter value, and generates a csv result file and convergence plots (c.f. next slide).

Results[1]

- F = objective function
- $F(x^*)$ = optimum
- $F(x) - F(x^*)$ = gap between current value (across iterations) and optimum



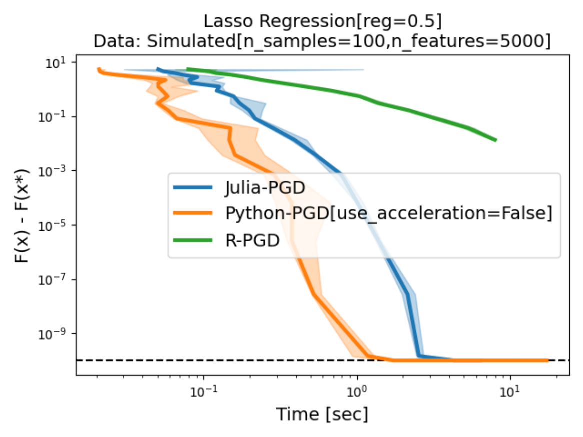


[1] Examples of standard benchmark results at `https://benchopt.github.io/results/`

Cross-language comparison

benchopt can also compare implementations in different languages.

Example comparing Proximal Gradient Descent (PGD) in Python, R, Julia



Available benchmarks

Data : $y \in \mathbb{R}^n, X = [x_{ij}] \in \mathbb{R}^{n \times p}$ with rows $x_i \in \mathbb{R}^p$

- ordinary least-squares (ols[2]) $\min_w \frac{1}{2}\|y - Xw\|_2^2$
- non-negative least-squares (nnls[3]) $\min_{w \geq 0} \frac{1}{2}\|y - Xw\|_2^2$
- l1-regularized least-squares (lasso[4]) $\min_w \frac{1}{2}\|y - Xw\|_2^2 + \lambda\|w\|_1$
- l2-regularized logistic regression (logreg_l2[5]) $\min_w \sum_i \log(1 + \exp(-y_i x_i^\top w)) + \frac{\lambda}{2}\|w\|_2^2$
- l1-regularized logistic regression (logreg_l1[6]) $\min_w \sum_i \log(1 + \exp(-y_i x_i^\top w)) + \lambda\|w\|_1$

[2] https://github.com/benchopt/benchmark_ols

[3] https://github.com/benchopt/benchmark_nnls

[4] https://github.com/benchopt/benchmark_lasso

[5] https://github.com/benchopt/benchmark_logreg_l2

[6] https://github.com/benchopt/benchmark_logreg_l1

Benchmark principle

A benchmark is a directory with:

- An `objective.py` file implementing an `Objective`
- A directory `solvers` containing different `Solver` implementation
- A directory `datasets` with `Dataset` generators/fetchers

Note: each objects above can be parametrized.

Possible to select the `objective/solver/dataset` you want to run.

Benchmark structure

```
my_benchmark/
├── README.rst
├── datasets
│   ├── simulated.py  # some dataset
│   └── real.py  # some dataset
├── objective.py  # contains the definition of the objective
└── solvers
    ├── solver1.py  # some solver
    └── solver2.py  # some solver
```

benchopt implementation

- benchopt core is written in Python
- Command Line Interface (CLI) to run benchmarks
- Python API to write benchmarks and add solvers (simple interface to call solvers implemented in other languages)
- Solver dependencies (both in Python and R) managed with conda[7].

[7] See https://docs.conda.io/projects/conda/en/latest/user-guide/install/

benchopt command

```
$ benchopt -h
Usage: benchopt [OPTIONS] COMMAND [ARGS]...

  Command-line interface to benchOpt

Options:
  -v, --version  Print version
  -h, --help     Show this message and exit.

Commands:
  clean    Clean the cache and the outputs from a benchmark.
  config   Configuration helper for benchopt.
  plot     Plot the result from a previously run benchmark.
  publish  Publish the result from a previously run benchmark.
  run      Run a benchmark with benchopt.
  test     Test a benchmark for benchopt.
```

Adding a solver to an existing benchmark

- Example of a standard solver (implemented in Python) in `benchmark_lasso/solvers/python_pgd.py`
- Example of a solver implemented in an R package (with a simple interface in Python) in `benchmark_lasso/solvers/glmnet.py`
- Example of a solver implemented in a local R file in `benchmark_lasso/solvers/r_pgd.R` with its interface in Python in `benchmark_lasso/solvers/r_pgd.R`

Writing a complete benchmark

Procedure detailed at `https://benchopt.github.io/how.html`

1. Implement an objective corresponding to an optimization problem in the file `my_benchmark/objective.py`[8]
2. Implement dataset simulator(s) and/or existing dataset fetcher(s) in the directory `my_benchmark/datasets`[9]
3. Implement solver(s) (with local implementation or by importing existing libraries) in the directory `my_benchmark/solvers`[10]

[8] Example for Lasso in `benchmark_lasso/objective.py`

[9] Examples for Lasso in `benchmark_lasso/datasets`

[10] Examples for Lasso in `benchmark_lasso/solvers`

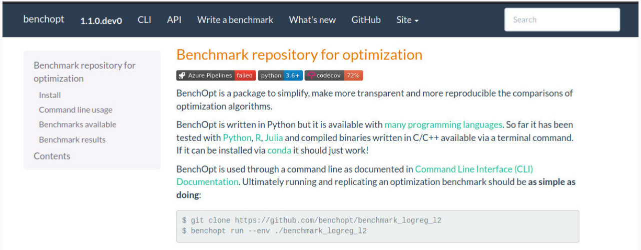
Managing the Python interface

!!! Adding solvers to benchmarks or writing complete benchmarks requires some knowledge of Python programming and object-oriented programming !!!

- Recommended to use an existing solver file or benchmark directory to create your own
- Example of `glmnet` solver in benchmark_lasso/solvers/glmnet.py

Resources

- Website: `https://benchopt.github.io/`



Benchmark repository for optimization

BenchOpt is a package to simplify, make more transparent and more reproducible the comparisons of optimization algorithms.

BenchOpt is written in Python but it is available with many programming languages. So far it has been tested with Python, R, Julia and compiled binaries written in C/C++ available via a terminal command. If it can be installed via conda it should just work!

BenchOpt is used through a command line as documented in Command Line Interface (CLI) Documentation. Ultimately running and replicating an optimization benchmark should be **as simple as doing**:

```
$ git clone https://github.com/benchopt/benchmark_logreg_l2
$ benchopt run --env ./benchmark_logreg_l2
```

- Development platform: `https://github.com/benchopt/benchOpt`

- Default benchmarks available at `https://github.com/benchopt`

WANTED

YOU

Join our effort to create reproducible benchmarks by adding new objectives/solvers/datasets!!!

Thanks for your attention

Questions ?