Optimization Algorithms
Coding Assignment 4

Joaquim Ortiz-Haro & Danny Driess & Marc Toussaint
Learning & Intelligent Systems Lab, TU Berlin
Marchstr. 23, 10587 Berlin, Germany

Winter 2022/23

The third coding assignment includes three tasks:

a) Compute the gradient of the solution of a parametric Quadratic Program (45 %)

b) Evaluate the impact of the initial step size and decay rate in stochastic gradient descent (25 %)

c) Implement the ADAM algorithm for stochastic optimization (30 %)

Deadline: Wednesday 08.02.2023 at 11.55 pm.

To work on the assignment, first check that the remote `upstream` is pointing to our repository.

```
git remote -v
```

You should see two remotes, similar to:

```
origin        git@git.tu-berlin.de:joaquimortizdeharo/optimization_algorithms_w22.git (fetch)
origin        git@git.tu-berlin.de:joaquimortizdeharo/optimization_algorithms_w22.git (push)
upstream        https://git.tu-berlin.de/lis-public/optimization_algorithms_w22 (fetch)
upstream        https://git.tu-berlin.de/lis-public/optimization_algorithms_w22 (push)
```

Otherwise, follow the instructions in `coding-readme.pdf`.

Now, you can merge our repository with your fork. If you have only modified the files in `assignments/a1_###`, `assignments/a2_###` and `assignments/a3_###`, the merge should be automatic (git will use the recursive strategy and it will ask for a commit message). Otherwise, you have to make sure that, after the merge, all files outside `assignments/a1_###`, `assignments/a2_###` and `assignments/a3_###`, are exactly the same as in our repository.

```
git fetch upstream
git merge upstream/main
```

1 Differentiable Optimization

Implement the optimization problem

$$\min_{x \in \mathbb{R}^2} \; c^T y^*(x) \;, \tag{1}$$

where $y^*(x)$ is the solution of the parametric Quadratic program (QP) (2)

$$\min_{y \in \mathbb{R}^2} ||y - x||^2 \text{ s.t. } Ay \leq b \;, \tag{2}$$

and $c \in \mathbb{R}^2$, $A \in \mathbb{R}^{4\times 2}$, and $b \in \mathbb{R}^4$ are given input parameters.
Hints and suggestions:

- For an input x, you could get $y^*(x)$ solving the QP (2) using an algorithm for constrained optimization, e.g. Augmented Lagrangian. To simplify the exercise, solving the QP is not required. Instead, you can query an oracle that provides the mapping $x \mapsto (y^*(x), \lambda^*(x))$ for a limited set of x. λ^* are the Lagrange multipliers of (2) at the optimum y^*.
- You should compute the gradient $\frac{d}{dx} y^*(x)$ using the optimality KKT conditions and the implicit function theorem. Assume that $y^*(x)$ is differentiable, i.e. the constraint activity does not switch.
- You should only implement the cost function and the gradient (but not the Hessian) of (1), using a single feature of type `OT.f`.

You will find some comments in `solution.py` to guide you in the implementation. Run and test your code with

```
cd assignments/a4_diff_opt
python3 test.py
python3 play.py
```

Once you are done, stage, commit and push `assignments/a4_diff_opt/solution.py`.

2 Solver: Stochastic Gradient Descent

Implement stochastic gradient descent $x' = x_k - \alpha_k \nabla f_i(x_k)$ using a learning rate with decay,

$$\alpha_k = \frac{\alpha_0}{(1 + \alpha_0 \lambda k)}, \tag{3}$$

where α_0, and λ are user-defined constants.
The input of the `solve` function is an object of a new class called `NLP_stochastic`, that represents a stochastic optimization problem of the form,

$$\min_x f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x) \tag{4}$$

Importantly, you can only query the function value and gradient of one index at a time, i.e. $f_i(x), \nabla f_i(x)$ for some chosen i. `NLP_stochastic.evaluate_i` has two inputs: `x` (the variable) and `i` (the index of the cost function you want to query), and returns the value and gradient of $f_i(x)$. `NLP_stochastic.getNumSamples()` returns the value of N.

In your implementation, you need to choose an appropriate α_0 and λ, and a strategy/ordering to choose which index to query next. The solver should fulfill the following requirements:

- The number of calls to `NLP_stochastic.evaluate_i(x,i)` is limited to 10.000. The solver should return a solution before reaching the maximum number of queries. Using the number of queries as termination criteria is a valid solution.
- The returned solution x^* is close to the real optimum, with $||x^* - x_{\text{opt}}|| \leq 0.05$.

```
cd assignments/a4_sgd
python3 test.py
python3 play.py
```

Once you are done, stage, commit and push `assignments/a4_sgd/solution.py`.

3 Solver: Adam

Implement the ADAM Solver (see Lecture Slides - `Stochastic Gradient Descent`) to solve stochastic optimization problems of the form:

$$\min_x f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) \tag{5}$$

The input of the `solve` function is an object of a new class called `NLP_stochastic` (see Section 2).
The solver should fulfill the following requirements:

- The number of calls to `NLP_stochastic.evaluate_i(x,i)` is limited to 10.000. The solver should return a solution before reaching the maximum number of queries. Using the number of queries as termination criteria is a valid solution.
- The returned solution x^* is close to the real optimum, with $||x^* - x_{\text{opt}}|| \leq 0.05$.

```
cd assignments/a4_adam
python3 test.py
python3 play.py
```

Once you are done, stage, commit and push `assignments/a4_adam/solution.py`.