Week 3 tasks

July 23, 2021

1. I think this week is a good time to address our so-called technical debt. What I mean is that we should invest some time in ensuring our code is clean, modular and well-commented as this will make our lives easier going forward. Specifically:

   (a) Do we want to modify the BaseOptimizer class at all? I can't think of anything immediately, but perhaps we want to adapt this to comparison-based optimization a bit.

   (b) Let's clean up the folder structure a bit. I think STP and SignOPT (in `SignOPT2.py`) are more-or-less in their final form, so lets put them in a folder called "Algorithms". We can keep algorithms that are still in progress in "ExampleCode". Also, lets separate out the functions used for benchmarking and plotting.

   (c) There are a lot of sub-routines common to all or most of the algorithms we've considered/ will consider. We should avoid rewriting these for every algorithm (to save time, but more importantly if there are subtle differences in the way each or these subroutines is implemented it will make our future benchmarking less valid). Let's put all of these in a file called `utils.py`, from which we can import them for each algorithm. As an example, most of our algorithms generate a matrix Z whose rows are the random sampling directions. We can make this a function, and have it take as input the number of rows, length of row and type (*i.e.* Gaussian, uniform from sphere and so on). Another example is using the comparison oracle to determine the best point among three or more (which is used in STP and GLD).

(d) Finally, we have quite a few git branches going on. I've merged the branch I was working on yesterday (and during our meeting) into the master branch, so I think "master" is the most up to date. Going forward, lets create one new branch per week to work on the week's tasks, and then merge it into "master" after our weekly meetings.

2. Write GLD as a class.

3. Write SCOBO as a class. I've added a Jupyter notebook to "Example-Code" containing a more vanilla implementation of SCOBO (*i.e.* one not tailored to MuJoCo). You may find it easier to work from this. Let me know if you have any questions! Some parts of this algorithm are tricky.

4. Benchmark GLD, STP, SignOPT and SCOBO on the three functions we have.

5. *(Longer-term)* There are a few recent papers explicitly dealing with comparison-based optimization. For example [KNT21, SKM21]. We should definitely look into these.

6. *(Longer term)* Our meeting on Thursday raised the very interesting possibility of considering comparison-based optimization allowing more than two feedback options, for example $1 = \{f(y) \text{ is better than } f(x)\}$, $-1 = \{f(x) \text{ is better than } f(y)\}$ and $0 = \{f(y) \text{ and } f(x) \text{ are roughly equal}\}$. We should think further on this.

7. *(To be done by Daniel).* I think the time has come for us to expand our collection of test functions. I will add this library to our project and make sure it interfaces well with our comparison oracle.

References

[KNT21] Mustafa O Karabag, Cyrus Neary, and Ufuk Topcu. Smooth convex optimization using sub-zeroth-order oracles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3815–3822, 2021.

[SKM21] Aadirupa Saha, Tomer Koren, and Yishay Mansour. Dueling convex optimization. In *International Conference on Machine Learning*, pages 9245–9254. PMLR, 2021.