Daily Log

Aryan Dua

June-July 2022

1 Day 1 - 13/06/22

- Saw YouTube videos about convexity and duality, mathematical optimisation. Namely - Convex Optimisation 1 by Stephen Boyd, Convex Optimisation Basics, The Hessian Matrix - Definition and Worked Example, Convexity and the principle of duality, What is mathematical optimisation? KKT Conditions and the interior point method, the proximal operator
- Went through webpages regarding convex optimisation

2 Day 2 - 14/06/22

- Saw YouTube videos about general optimisation and quasiconvex functions solved a few basic optimisation problems. Made notes about a few important pointers. The videos watched were: Understanding quasiconcave and quasiconvex functions, Linear Algebra - Projection onto surfaces
- Went through the julia language syntax

3 Day 3 - 15/06/22

- Wrote the math to figure out the projection operators on 3 kinds of sets. N-dimensional balls, n-dimensional equations represented by A'X = b, n-dimensional half-planes represented by

$$A'X \leq b$$

- Wrote a running julia code for the above task

4 Day 4 - 16/06/22

- Wrote a running julia code that runs a constrained gradient descent program, when given a function f(x) = x' * A * x + b' * x, and a given domain set.

5 Day 5 - 17/06/22

- Finished the initial set of homework problems given in notes.pdf, wrote the solutions in my notebook. Researched about the interesting parts of the questions while going through them.
- Attended the seminar by Jonathan Eckstein

6 Day 6 - 20/06/22

- Tried to plot my gradient descent program using a projection operator in Julia.
- Had a progress check done with Zev at 2pm
- Started reading about asynchronous programming, and the abstract of the Comb18 research paper.

7 Day 7 - 21/06/22

- Read about the principle of duality. Watched this video to understand duality better : Bierlaire (2015) Optimization: principles and algorithms, EPFL Press. Section 4.1
- Researched more about asynchronous programming.
- Mathieu explained duality and block-iterativeness.

8 Day 8 - 22/06/22

- Had a 2 hour session with Zev at 2pm. Discussed the mathematical definition of gradient, learned about subgradients an subdifferentials. Started to understand Algorithm 4, to start solving our problem.

9 Day 9 - 23/06/22

- Learned how to use indicator functions and prox operators in Julia, and installed the required libraries.
- Spent time decoding the notation of the research paper, under algorithm 4.
- Tried to run a tried and tested code by Douglas and Rachford.

10 Day 10 - 24/06/22

- Made a note of how I am to proceed with the coding part of the algorithm, noted my approach and doubts.
- Had a meet with Zev and cleared the notation and algorithmic doubts. Here are the highlights of the session: 1) gamma and mu are random sequences of numbers, their values lie between epsilon and 1/epsilon. 2) K is the part with the linear operators, i.e. parts which are not in I. 3) Parts in I include the functions with ONLY 1 variable. 4)

11 Day 11 - 27/06/22

- Made the first draft of the running program in Julia
- Tried to tune the hyperparameters to give an optimal solution but ended up with a convergent solution, on the wrong side of the domain.

12 Day 12 - 28/06/22

- Exported the important variables from the julia file to a text file and then imported them into a python notebook
- Made a plot for the optimisation, tried a few heuristics with the hyperparameters I could tune.

13 Day 13 - 29/06/22

- Solved the problem of the final result staying out of the given bounds, but faced a new problem - what exactly are the dual solutions and what is their significance in this algorithm?
- Experimented with the parameters further.

14 Day 14 - 30/06/22

- Had a meeting with Zev to discuss further doubts. Made some corrections to the previous algorithm to have our first working draft ready. Discussed the next steps, and got a homework question - visualise the conjugate of a function.

15 Day 15 - 04/07/22

- Converted the code into a much more structured, organised and readable format.
- Made the number of manual inputs required less, and separated the coding area into the manual input part, and the algorithm part, where it derives all the required variables from the few inputs defined above.

16 Day 16 - 05/07/22

- Made the code work for any number of K's. The current state of the algorithm is that it can implement any number of indicator functions in 1 variable(x1).

17 Day 17 - 06/07/22

- Incorporated the L matrix into the program, so the algorithm can compute functions like x1-x2 into the "g" part of the minimization problem
- Had a meeting with Zev to discuss the implementation of the non-zero slope function. The current state of the algorithm is that it works for any number of vectors in I and K, and for any coefficients in L.

18 Day 18 - 07/07/22

- Tested the algorithm on new functions. The latest test included minimizing $x_1 - x_2$ subject to the constraint of a point circle, and the intersection of 2 sets of 2 circles of radius 1 unit each, and it worked out as expected.

19 Day 19 - 08/07/22

- Had a meeting with Zev and Mathieu. Mathiee explained the working of an asynchronous algorithm in Julia. Zev explained the new tasks that I will have to complete, which include - Making Minibatches work, taking "L" as either a matrix or a Linear Operator Function and finally, making the algorithm work asynchronously.

20 Day 20 - 12/07/22

- Worked out the logic for selecting only a few Is and Ks in a single iteration such that, by every M iterations all Is and Ks have been used at least once. The logic was implemented using a random bit vector in each iteration.

21 Day 21 - 13/07/22

- Completed the asynchronous implementation of the algorithm. It was done using an array of tasks and their birth iterations and properly fetching and scheduling them as needed. I included the conditions for normal and forced fetches as well.

22 Day 22 - 14/07/22

- The algorithm now accepts the L variable to be a matrix as well. So instead of multiplying x1 and x2 by 1s and 0s, I can now do it by the identity/null matrices (and so on)

23 Day 23 - 15/07/22

- Implemented the array implementation of mu and gamma to incorporate the asynchronous factor into them.
- Had a meet with Zev and Mathieu to discuss the current progress and the tasks for the future. Mathieu suggested I use more functions in my code.
- Mathieu also sent a link on how to use threads in the program.
- The further tasks discussed were to somehow introduce an artificial sleep in one of the proxes, say i = 1. This was to be done to check if the forceful fetch condition of the tasks is met or not. If yes, then the asynchronous implementation has been done correctly.

24 Day 24 - 18/07/22

- Figured out how to use threads. I am currently using 4 threads, as suggested by Mathieu.
- Organised the code into 2 files- async-prox.jl and functions.jl. The functions file contained all the functions in the program.

25 Day 25 - 19/07/22

- I tested the program with different functions today. Had surprisingly bizarre results. When I ran the program directly, it gave me a big error margin, but if I printed along each iteration, or if I added a 1 millisecond sleep in each iteration, I got a perfect answer.
- I discussed the above observation with Mathieu and he suggested that maybe the problem was due to the fact that my problem had a single

point solution, and since the values outside the constrained area is +inf, it would be hard to optimise to exactly that point.

- I also had a meet with Zev. We discussed the future applications of the algorithm and next tasks assigned were to return the feasibility of the solution and plots of the iterate.

26 Day 26 - 20/07/22

- I implemented L as function. Now I can input L as either a matrix or matrices(a bigger matrix), a vector of functions or a vector of numbers.
- I also organised the code into 5 different julia files Saved the program output into 2 files x1.txt and x2.txt. Made a jupyter notebook to plot the graphs of the iterate and also a README for a better understanding of the project.
- Finally, I uploaded everything to Github from the command line.