Daily Log

Aryan Dua

June 13 2022 - July 29 2022

1 13 June 2022

- While researching online I came across the following videos and found them quite informative:
  - 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
- Spent some time doing online research on Convex Optimization and gathered a few useful insights.

2 14 June 2022

- Researched online regarding the following topics:
  - Understanding quasiconcave and quasiconvex functions
  - Linear Algebra - Projection onto surfaces
- Solved optimisation problems that I encountered online.
- Went through the Julia language syntax

3 15 June 2022

- Worked out the formulae to figure out the projection operators on 3 kinds of sets.
  1. N-dimensional balls represented by $||x|| <= r$

2. N-dimensional equations represented by $A'X = b$
3. N-dimensional half-planes represented by $A'X \leq b$

- Successfully compiled a working Julia code for the above task.

4 16 June 2022

- Successfully tested a Julia code that runs a constrained gradient descent program, when given a function: $f(x) = x^\top \cdot A \cdot x + b^\top \cdot x$ and a given domain set.

5 17 June 2022

- Finished the assigned tasks given by the project guide ahead of time. A few topics caught special attention and that made me do extensive research on them.
- Attended the seminar by Jonathan Eckstein on the topic "Solving Stochastic Programming Problems by Operator Splitting"

6 20 June 2022

- Worked on plotting the gradient descent code that I had written using a projection operator in Julia.
- Went through some articles regarding asynchronous programming, and the abstract of the Comb18 research paper.

7 21 June 2022

- Studied the "Principle of Duality". Watched this online video to understand duality better : Bierlaire (2015) Optimization: principles and algorithms, EPFL Press. Section 4.1
- As advised by the project guide, I read more about Asynchronous programming.
- My supervisor explained Duality and Block-Iterativeness.

8 22 June 2022

- Had an interactive and fruitful session with my guide on the following topics:
  - The Mathematical definition of a Gradient

  - Subgradients and Subdifferentials
  - Algorithm 4 in the Comb18 research paper - The implementation of which is the task of my internship.

9 23 June 2022

- Learnt how to use indicator functions and proximal operators in Julia, and installed the required libraries.
- Spent time decoding the notation of the research paper, under algorithm 4.
- I was lucky enough to find a ready to use code based on the algorithm by Douglas and Rachford. I created a few additional test scenarios to make sure the code works in all conditions.

10 24 June 2022

- Reviewed my progress and worked out the future course of action.
- Had a doubt clarification session with my supervisor about the notation and algorithms. The following topics were covered:
  - Gamma and Mu are random sequences of numbers
  - Their values lie between epsilon and 1/epsilon
  - K is the part with the linear operators, i.e. parts which are not in I.
  - Parts in I include the functions with ONLY 1 variable.

11 27 June 2022

- After a lot of study and analysis, I arrived at the first draft of the running program of algorithm 4 in Julia
- Worked on tuning the hyperparameters to give an optimal solution. The result was quite close to the expected output.

12 28 June 2022

- 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, and tuned the hyperparameters using heuristics.

13 29 June 2022

- Although it took some time, but the final outcome was worth every minute of it.
- The next task was to work on the following -
  - What exactly are the dual solutions?
  - What is their significance in this algorithm?
- Experimented with the parameters further.

14 30 June 2022

- Reviewed and redrafted the previous algorithm and created the first working draft.
- Had a doubt clarification session with my supervisor. Discussed the next steps, and got another assignment - visualise the conjugate of a function.

15 1 July 2022

- Watched online videos on the following topics:
  - Conjugate functions I: Definition and properties, by Sebastian Banert.
  - Dual Optimal Solutions - Georgia Tech - Complexity, Theory, Computability By Udacity

16 4 July 2022

- Further modified the code to make it better structured, more organised, and more readable by adding some comments.
- Optimised and organised the code into the manual input part and the algorithmic part eliminated the redundancy.

17 5 July 2022

- Made the code work for any number of K's. At this stage, the algorithm could implement any number of indicator functions in 1 variable (x_1).

18 6 July 2022

- Introduced the L matrix into the program, so the algorithm could compute functions like $x_1 - x_2$ into the "g" part of the minimization problem.
- Had a session with my supervisor to figure out the best implementation approach for the non-zero slope function. At this stage, the algorithm could work for any number of vectors in I and K, and for any coefficients in L.

19 7 July 2022

- 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.

20 8 July 2022

- Had a session with my supervisors. They explained the working of an asynchronous algorithm in Julia, and the new tasks that I had to complete were -
  - Making minibatches work
  - Taking "L" as either a matrix or a Linear Operator Function
  - Making the algorithm work asynchronously.

21 11 July 2022

- Read the Julia documentation about using threads and making asynchronous code using tasks.
- Got a good grasp over the syntax of the commands used.
- Thought of a logic to work out the minibatches, such that both the conditions under Assumption 3 (a terminology from the research paper) were followed. The conditions were:
  - Every I and K should be activated in the first iteration.
  - There exists a number M, such that after every M iterations, every I and K must have been activated.

22 12 July 2022

- 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 had been used at least once. The logic was implemented using a random bit vector in each iteration.

23 13 July 2022

- 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.

24 14 July 2022

- Added some functionality such that the algorithm accepts the L variable to be a matrix as well. So instead of multiplying x_1 and x_2 by 1s and 0s, I could do it by the identity/null matrices (and so on)

25 15 July 2022

- Implemented the array implementation of mu and gamma to incorporate the asynchronous factor into them.
- Had a session with my supervisors to discuss the current progress and the tasks for the future. They suggested I use more functions in my code.
- My supervisor advised me 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 was met or not. If yes, then the asynchronous implementation had been done correctly.

26 18 July 2022

- Figured out how to use threads. At this stage, I was using 4 threads, as advised by my supervisor.
- Organised the code into 2 files - main.jl and functions.jl. The functions file contained all the functions in the program.

27 19 July 2022

- Tested the program with different functions. 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.
- Discussed the above observation with my supervisor 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.

- Had a session with my supervisor and we discussed the future applications of the algorithm and the next tasks assigned were to return the feasibility of the solution and plots of the iterate.

28 20 July 2022

- I implemented L as function, and therefore, I could 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.

29 21 July 2022

- Wrote a function that returned true if the the optimised result calculated from the algorithm was feasible, i.e. lying in the domain(does not blow up)
- Made the code more user-friendly, and easier to understand.

30 22 July 2022

- Made a few test functions to test the reliability of the code. The functions that I managed to optimise were, finding the minimas and maximas of either coordinates of the intersection of 2 circles in R^2
- This was working on most of the tests I ran. But, I noticed in 4 out of the 16 cases of ones and zeros, that it was not converging to a solution.
- Arranged for a virtual session with my supervisor to discuss this problem with him.

31 25 July 2022

- As my supervisor suggested, I compared each iteration of the sync version and the async version and noted my observations.
- The observations were that on making the test conditions similar by adjusting the parameters, the 2 outputs were quite similar up to the 4^{th} iteration, and then there was a difference in the 14^{th} decimal digit, which blew up by the 10000^{th} iteration to give a final difference of about 0.1.

32 26 July 2022

- It was still not converging only in those 4 cases.
- Had a session with my supervisors, and discussed the non-convergence error in the 4 specific cases.

33 27 July 2022

- Made some changes in the code of the sync program to plot the graphs of the sync version in the same Jupyter notebook as well.
- Printed the norm of (x v^*) as output, and also plotted the graph of this norm in the Jupyter notebook.
- Edited the Python code so that all the plots were visible at once. (Used subplots)

34 28 July 2022

- Spent time matching the code with the algorithm to see if there were any logical errors in the code.
- I also checked the problems with the scopes of the variables - problems arising due to the global and local variable declarations

35 29 July 2022

- Experimented with a few values of gamma, mu, alpha, beta, D and found out that by changing the values of gamma mu and alpha, the non-converging cases also converged.
- The understanding of the problem based on the above observation was that in the case of these 4 inputs, a larger "push" was required initially, which was why a very high value of lambda favoured the optimal result.

- Could finally conclude that the code was semantically correct as per the algorithm given in the research paper, and the big task at hand was to find a reliable hyperparameter selection strategy.