Project Guide for Notions of Positivity Project

Written by Karim El-Sharkawy of Purdue Mathematics

This document intends to provide a simple map for anyone wanting to familiarize themselves with this project. This can be used for review or as a guide. I am working on this project with Darshini Rajamani (LinkedIn), Abbas Dohadwala (LinkedIn), and Luke Luschwitz (LinkedIn). Professor Thomas Sinclair (website) of Purdue Mathematics supervises this project.

The GitHub to the project is linked below, where you can also access the GitHubs of all the contributors

https://github.com/karim-sharkawy/Notions-of-Positivity-and-Complexity-in-Quantum-Information-Theory

An understanding of linear algebra concepts (norm, isomorphism, etc) will be needed before tackling the theory of this project. Experience in a proof-based linear algebra course is helpful as well as going over the Professor's notes. Some concepts in the professor's notes are expanded upon below. It's good to learn some of the new material as you go so it's not overwhelming in the beginning.

Before going over the notes, it's important to know what a positive matrix is

+ What Does It Mean For a Matrix to be POSITIVE? The Practical Guide to Semide…
+ The Practical Guide to Semidefinite Programming (2/4)

Important topics to read about:

+ Ordered Vector Spaces: https://en.wikipedia.org/wiki/Ordered_vector_space
    + Summary:
+ Preorders & Partial Orders: https://en.wikipedia.org/wiki/Preorder
    + Summary:
+ Extendability
    + Summary:
+ Positive Lifting
    + Summary:
+ Hyperplane

The use of Python packages is essential to understand the code. What you need to know:

1. NumPy: Python NumPy Tutorial for Beginners
    a. Follow along with the video: GitHub - KeithGalli/NumPy: Jupyter Notebook & Data Associated with my Tutorial video on the Python NumPy Library
2. SciPy: SciPy v1.11.4 Manual
    a. More specifically, we used scipy.optimize.linprog: scipy.optimize.linprog — SciPy v1.11.4 Manual
3. Matplotlib
    a. Matplotlib Full Python Course - Data Science Fundamentals
    b. Plot types — Matplotlib 3.8.2 documentation
4. Random: Used to generate random numbers
5. Machine learning packages will be explained later

Explaining the code

1. The first code block recursively creates a positive semidefinite matrix that satisfies the E(2,2), linearity, and positivity constraints
    a. Important note: Currently, we're only working on 4 x 4 matrices. The code will be generalized later for any N x N size
    b. The E(2,2) constraint checks that the sum of first two values of every row equal the sum of the last two values
    c. The linearity constraint checks that the sum of the first two rows equals the sum of the last two rows
    d. The positivity constraint checks if the matrix is positive. Please watch the videos referenced above for the definition of a positive matrix
        i. A matrix that isn't positive is discarded, since we need all matrices that are mapped to be positive
2. The second block checks for liftability (extendability in this case too), using the 'linprog' function in SciPy. How this works is that …
    a. We tell it how many mappings to create, and after creating all the mappings along with their identifications (extendable or non-extendable), it appends them to a list depending on whether they're extendable or not

   b. It then saves the lists as files. This is used to decrease the wait time and with testing
   c. ‘c’ is “The coefficients of the linear objective function to be minimized”. This is then equated to something, ‘mat’ in this case.
   d. ‘A_ub’ is “The inequality constraint matrix. Each row of A_ub specifies the coefficients of a linear inequality constraint on c” and ‘b_ub’ is “The inequality constraint vector. Each element represents an upper bound on the corresponding value of A_ub”
   e. ‘A_eq’ is “The equality constraint matrix. Each row of A_eq specifies the coefficients of a linear equality constraint on c” and ‘b_eq’ is “The equality constraint vector. Each element of A_eq @ x must equal the corresponding element of b_eq”
3. The next few blocks save the mappings into files, load the files again if need, then print the number of extendable and non-extendable mappings
4. The next block finds the farthest non-extendable matrix from the extendable matrices. This is done through the use of the norm function (np.linalg.norm)
5. We’ll look into the machine learning code shortly, make sure to understand what the code is doing and why it works. Check the last bullet point of the next section to see a summary of the ML code
6. Each block of code has the name of the contributors on it, please check the file to know who created what

Before explaining the rest of the code, it's important to have a basic understanding of how machine learning works

- There are different types of ways how the computer learns, we’ll be using the classification method
- Linear classifier: Linear classifier - Wikipedia
    - Summary:
- SciKit Learn
    - 1. Supervised learning — scikit-learn 1.3.2 documentation
    - Scikit-learn Crash Course - Machine Learning Library for Python

- The last block of code uses SciKit-learn. It …

Other random things

+ Farkas’ theorem
    + Farkas' lemma - Wikipedia
    + Proof: A NICE PROOF OF FARKAS LEMMA 1
    + Deutsch
    + This is used to prove …
+