

## Graded Sheet 1

Due: 3. June 2022

# Multiple state discrimination via semidefinite programming

## Some background on Semidefinite programming

John Watrous' [notes](#) for the course *Semidefinite Programming in Quantum Information*, [Lectures](#).

## CVX package for MATLAB

It can be downloaded from [here](#), and there is an introductory video from Stephen Boyd on how to use it [here](#).

The aim of this project is to familiarize ourselves with optimization techniques that have a wide range of applications in quantum information theory and beyond. As a main tool, we use semidefinite programs (SDP), which are a special case of the more general class of convex optimization problems. The nice property of SDPs is that, although the theory underlying the algorithms to solve these optimization problems is non-trivial (see [1, 2] as an example), there exist robust implementations of these algorithms that do not require knowledge about the underlying algorithms (except for highly complex problems). Thus, the only non-trivial problem is to recognize that the problem at hand is an SDP and then to use the provided software packages properly.

## 1 Introduction to semidefinite programs

In this section, you will learn what an SDP is. We will start with an example from quantum information theory. Suppose you have some quantum device, which outputs a quantum system  $\rho?$ . The device is set in such a way that the output system is described by a density matrix  $\rho_1$  with probability  $p$  and it is described by  $\rho_2$  with probability  $1 - p$ . You have this system at your disposal and you should decide if the system is described by  $\rho_1$  or  $\rho_2$ . Since  $\rho_1$  and  $\rho_2$  need not be orthogonal, you cannot decide this problem with certainty. The best you can do is to perform a measurement on the system and guess the state with the help of the information obtained via the measurement. Of course, you want to maximize the probability of guessing right. The probability of guessing right is given by the product rule as

$$P_{succ} = \mathbb{P}(\text{guess } \rho_1 | \rho? = \rho_1) \mathbb{P}(\rho? = \rho_1) + \mathbb{P}(\text{guess } \rho_2 | \rho? = \rho_2) \mathbb{P}(\rho? = \rho_2) \quad (1)$$

$$= p \mathbb{P}(\text{guess } \rho_1 | \rho? = \rho_1) + (1 - p) \mathbb{P}(\text{guess } \rho_2 | \rho? = \rho_2). \quad (2)$$

Since you decide, based on your measurement outcome, between two hypotheses, it suffices to have a measurement with two possible outcomes. Remember that the most general description of measurement is given by a positive operator valued measure (POVM). For the case of a measurement with two outcomes, such a POVM is fully described by two effect operators  $E_1$  and  $E_2$ . In order to be a valid choice,  $E_1$  and  $E_2$  have to satisfy the following relations:

$$E_1 \geq 0, \quad E_2 \geq 0, \quad E_1 + E_2 = \mathbb{1}, \quad (3)$$

where  $\mathbb{1}$  denotes the identity matrix. W.l.o.g, by Born's rule, we then have that  $\mathbb{P}(\text{guess } \rho_1 | \rho_? = \rho_1) = \text{tr}[E_1 \rho_1]$  and  $\mathbb{P}(\text{guess } \rho_2 | \rho_? = \rho_2) \mathbb{P}(\rho_? = \rho_2) = \text{tr}[E_2 \rho_2]$ . Hence, in order to choose the POVM such that  $P_{succ}$  is maximized, you encounter the following optimization problem:

$$\begin{aligned} \text{maximize: } & p \text{tr}[E_1 \rho_1] + (1 - p) \text{tr}[E_2 \rho_2] \\ \text{subject to: } & E_1 + E_2 = \mathbb{1} \\ & E_1 \geq 0, E_2 \geq 0 \end{aligned} \tag{4}$$

The optimization problem (4) is in the form of an SDP. More specifically, it is an optimization over Hermitian matrices with linear objective function, linear equality constraints and linear constraints that require that some matrix is positive semidefinite. For a general SDP, we simply allow for more than two variables, and more linear equality and inequality constraints.

**Definition. 1.1.** *An SDP is an optimization problem of the following form*

$$\begin{aligned} \text{minimize/maximize: } & f(E_1, E_2, \dots, E_N) \\ \text{subject to: } & g_1(E_1, E_2, \dots, E_N) = A_1 \\ & \vdots \\ & g_M(E_1, E_2, \dots, E_N) = A_M \\ & h_1(E_1, E_2, \dots, E_N) \leq B_1 \\ & \vdots \\ & h_K(E_1, E_2, \dots, E_N) \leq B_K \\ & E_1 \in H_{d_1}, E_2 \in H_{d_2}, \dots, E_N \in H_{d_N} \end{aligned}$$

To specify such an SDP we need positive integers  $d_1, d_2, \dots, d_N, a_1, a_2, \dots, a_M, b_1, b_2, \dots, b_K$ ; Hermitian matrices  $A_1, A_2, \dots, A_M, B_1, B_2, \dots, B_K$ , with  $A_m \in H_{a_m}$  and  $B_k \in H_{b_k}$  and **linear** maps  $f, g_1, g_2, \dots, g_M, h_1, h_2, \dots, h_K$ , with the following signatures  $f : \times_{n=1}^N H_{d_n} \rightarrow \mathbb{R}$ ,  $g_m : \times_{n=1}^N H_{d_n} \rightarrow H_{a_m}$  and  $h_k : \times_{n=1}^N H_{d_n} \rightarrow H_{b_k}$ .

**Remark. 1.**

- 1) One can replace the equality constraints by two inequality constraints. Furthermore, all inequality constraints can be combined into one function  $\times_{n=1}^N H_{d_n} \rightarrow \times_{k=1}^K H_{b_k}$ . Hence there is some redundancy in the definition above.
- 2) It follows from the Riesz-representation theorem that  $f$  can be written in the form  $f(E_1, E_2, \dots, E_N) = \sum_{n=1}^N \text{tr}[F_n E_n]$ , for some  $F_n \in H_{d_n}$ .
- 3) Valid constraints include
  - $E_1 \geq 0$ ,
  - $E_1 - E_2 + 4E_5 = A$ ,
  - $XE_1X^* - 5YE_1Y^* + ZE_2Z^* \leq 0$ ,
  - $\text{tr}[E_1] = 1$ ,

where  $X, Y, Z$  are (not necessarily Hermitian) matrices and it is assumed that the dimensions match, so that the expressions above make sense.

4) It can be shown that every constraint function can be written in the following form

$$g(E_1, E_2, \dots, E_N) = \sum_{n=1}^N \sum_{i=1}^{L_n} \sigma_{n,i} K_{n,i} E_n K_{n,i}^*, \quad (5)$$

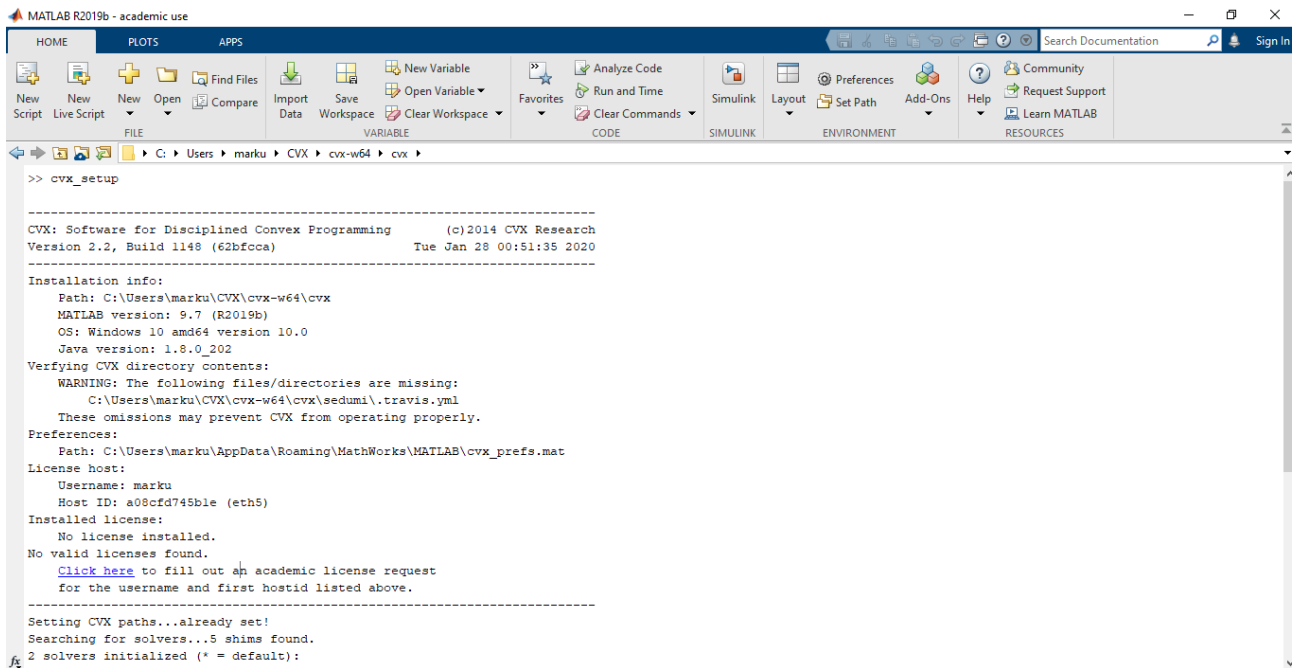
where  $K_{n,i}$  is a  $a \times d_n$  matrix (for some  $a \in \mathbb{N}$ ),  $\sigma_{n,i}$  are real coefficients and  $L_n \in \mathbb{N}$ .

## Exercise 1: Trace-norm

The optimal value of (4) has an expression in terms of the trace norm  $\|\cdot\|_1 := \text{tr}|\cdot|$ . Derive such an expression.

## 2 How to solve an SDP with MATLAB

In this section, you will learn how to use the `cvx` package in Matlab to solve SDPs. To install `cvx` in Matlab, [download the package under this link](#). Unpack the package, open Matlab and navigate to the obtained folder. Then type `cvx_setup`. Your screen should now look like this:



```

MATLAB R2019b - academic use
HOME PLOTS APPS
New Script New Live Script New Open Find Files Import Data Save Open Variable Clear Workspace
FILE VARIABLE
Analyze Code Run and Time Favorites Run and Time Clear Commands
SIMULINK LAYOUT Set Path Add-Ons Help Community Request Support Learn MATLAB
Search Documentation Sign In

C:\Users\marku\CVX\cvx-w64\cvx
>> cvx_setup

-----
CVX: Software for Disciplined Convex Programming (c)2014 CVX Research
Version 2.2, Build 1148 (62bfcca) Tue Jan 28 00:51:35 2020
-----
Installation info:
  Path: C:\Users\marku\CVX\cvx-w64\cvx
  MATLAB version: 9.7 (R2019b)
  OS: Windows 10 amd64 version 10.0
  Java version: 1.8.0_202
Verifying CVX directory contents:
  WARNING: The following files/directories are missing:
  C:\Users\marku\CVX\cvx-w64\cvx\sedumi.travis.yml
  These omissions may prevent CVX from operating properly.
Preferences:
  Path: C:\Users\marku\AppData\Roaming\MathWorks\MATLAB\cvx_prefs.mat
License host:
  Username: marku
  Host ID: a08cfd745b1e (eth5)
Installed license:
  No license installed.
No valid licenses found.
  Click here to fill out an academic license request
  for the username and first hostid listed above.
-----
Setting CVX paths...already set!
Searching for solvers...5 shims found.
2 solvers initialized (* = default):
  
```

Figure 1: Cvx package for Matlab.

It's also recommendable that you watch the `cvx` introduction video by Stephen Boyd, which you can find [here](#). We are now ready to solve the first SDP in (4). The Matlab code in Figure 2 solves the problem.

We will go through this code line by line. In the first four lines, we initialize the variables that describe the problem. In our case, we try to discriminate between the maximally mixed state  $\rho_1 = \frac{1}{2}\mathbb{1}$ , which appears with probability  $p = 0.4$  and the state  $\rho_2 = |0\rangle\langle 0|$ . A semi-definite program is described between the two lines `cvx_begin` and `cvx_end`. The expression `sdp` in line 5 tells `cvx`

```

Command Window
1 >> p = 0.4;
2 dim = 2;
3 rho1 = [.5, 0]; [0, .5];
4 rho2 = [1, 0]; [0, 0];

5 cvx_begin sdp quiet
6 variable E1(dim,dim) hermitian;
7 variable E2(dim,dim) hermitian;

8 maximize ( real( p*trace(E1*rho1) + (1-p)*trace(E2*rho2) ) );
9 E1 >= 0;
10 E2 >= 0;
11 E1+E2 == eye(dim);

12 cvx_end
>> cvx_optval

cvx_optval =
    0.8000                                Optimal value

>> E1
E1 =
    0.0000 + 0.0000i    0.0000 - 0.0000i
    0.0000 + 0.0000i    1.0000 + 0.0000i

>> E2
Optimal measurement operators
E2 =
    1.0000 + 0.0000i    -0.0000 + 0.0000i
   -0.0000 - 0.0000i    0.0000 + 0.0000i

```

Figure 2: Matlab code to solve the SDP in (4).

that inequalities are meant in the Loewner sense and `quiet` suppresses the output of the protocol of the calculation. In lines 6 and 7, we declare two variables, the  $\text{dim} \times \text{dim}$  hermitian matrices  $E_1$  and  $E_2$  (these are the measurement operators). In the same way you can declare any number of matrices. You might want to declare  $n \in \mathbb{N}$  Hermitian matrices of size  $d \times d$ . Then the syntax would be `variable E(dim, dim, n) hermitian;` and you can access the  $k$ -th variable by `E(:, :, k)`.

```

1
2     variable E(dim, dim, n) hermitian;
3
4
5     temp = zeros(dim,dim);
6     for k = 1:n
7         temp = temp + E(:, :, k);
8     end
9     temp >= 0;

```

In lines 8-11, we declare our problem, which is that we want to maximize the expression  $p \text{tr}[E_1 \rho_1] + (1-p) \text{tr}[E_2 \rho_2]$ . Taking the real part of this expression is not necessary mathematically, since the expression is always linear. However, due to rounding errors the real part might be negligibly small, but non-zero (in which case `cvx` might output an error message). In lines 9 and 10 we demand that  $E_1$  and  $E_2$  are positive semi-definite. The `cvx` framework allows for Loewner inequalities of the form given in (5). For example, for two arbitrary  $b \times \text{dim}$  matrices  $X$  and  $Y$  and a Hermitian  $b \times b$  matrix  $B$ , `X*E1*ctranspose(X) - 5*Y*E2*ctranspose(Y) <= B` would be a legal syntax for a constraint. You can also use loops to generate the objective function or to declare constraints. For example, you might want to write a code similar to the one below, if you want to state the constraint that  $\sum_{k=1}^n E_k$  should be positive semi-definite.

Finally, in line 11, we state that  $E_1 + E_2 = \mathbb{1}$ . Finally, by the command `cvx_end`, the calculation starts. The optimal value is then stored in the variable `cvx_optval` and the declared matrices are

then ordinary Matlab matrices such that the optimum is attained.

### 3 Multiple state discrimination via semidefinite programming

This task is a generalization of the previous introductory example. Given a set of density matrices  $\{\rho_i\}_{i=1}^N$  and a quantum state  $\rho$ , we are promised that  $\rho$  is in each state  $\rho_i$  with probability  $p_i$ , so that  $\sum_{i=1}^N p_i = 1$ . The aim of this query is to successfully identify in which of the states  $\rho_i$ , lies the state  $\rho$ . This problem is known as the *quantum hypothesis testing*.

Let  $N$  denote the number of quantum states we consider, i.e., for each  $1 \leq i \leq N$ ,  $\rho_i \in \mathcal{M}_{d \times d}(\mathbb{C})$  with  $\rho_i \geq 0$  and  $\text{tr}[\rho_i] = 1$ . Moreover, assume that we are given a probability distribution  $\{p_i\}_{i=1}^N$ . Following the idea presented in the introductory example, the goal here is to optimize the value of  $\sum_{i=1}^N p_i \text{tr}[\rho_i E_i]$  over POVMs (Positive-Operator Valued Measures)  $\{E_i\}_{i=1}^N$ , i.e.,  $E_i \geq 0$  for  $i \in \{1, 2, \dots, N\}$  and  $\sum_{i=1}^N E_i = \mathbb{1}$ .

#### Exercise 2: $N = 3$ and pure states

Assume that we consider three quantum states  $\rho_i$ , i.e.  $N = 3$ , that are pure and orthonormal to each other. Write down and justify the best POVM you can choose.

#### Exercise 3: State discrimination via SDP

Write a program that computes the following supremum:

$$\sup_{\{E_i\}_{i=1}^N \text{ POVM}} \sum_{i=1}^N p_i \text{tr}[\rho_i E_i]. \quad (6)$$

using an SDP.

Given the previous  $\rho_i$  and  $p_i$  for  $i \in \{1, \dots, N\}$ , the first approximation of a POVM would be defined by  $\bar{E}_i = p_i \rho_i$ , for all  $i$ . However, this is not a POVM because  $S = \sum_{i=1}^N \bar{E}_i \neq \mathbb{1}$  (check that  $\text{tr}[S] = 1$ ).

#### Exercise 4: Pretty good measurement

For every  $i \in \{1, \dots, N\}$ , define  $E_i := S^{-1/2} p_i \rho_i S^{-1/2} + \frac{1}{N} (\mathbb{1} - S^{-1/2} S S^{-1/2})$ . Show that  $E_i$  is a POVM (There is a general way to give a sort-of-inverse for all matrices, even if they are not square or of full rank: Moore-Penrose pseudoinverse).

### Exercise 5: $N = 3$ and, the optimal POVM is not the pretty good measurement

Consider again the case  $N = 3$ , the following three Hermitian matrices:

$$\rho_1 := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \rho_2 := \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \quad \rho_3 := \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad (7)$$

and the probabilities  $p_1 = p_2 = p/2$  and  $p_3 = 1 - p$ . Show that the optimal POVM is not the pretty good measurement. Furthermore, compare the probability of success while using the optimal POVM with the one of the pretty good measurement, in a graph.

*Hint: It might be hard to find an analytical solution to this exercise, so in order to find the graph, you need to solve an SDP for different values of  $p$ .*

## References

- [1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [2] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, [link](#), 1996.