EE5121 Convex Optimization
CVX Assignment

Rajat Vadiraj Dwaraknath
EE16B033

May 8, 2019

1 Introduction

This report contains the solutions to the CVX assignment of the course EE5121: Convex Optimization. The problems are solved in python, using the python equivalent of CVX in Matlab - the `cvxpy` package. Problem 1 is solved using two approaches, with one python script for each method, while the remaining two problems have one python script each as well.

2 Question 1: Piecewise Constant Recovery

This problem was solved using two approaches. First, is the standard approach of approximating the cardinality of a vector via the ℓ_1 norm relaxation, and formulating the regularized problem as a second-order cone program. The second method enhances sparsity using a method called the reweighted-ℓ_1 minimization algorithm.

2.1 SOCP formulation

The code for this approach can be found in `ee16b033_q1.py`.

Using the terms defined in the question, we can formulate our final objective as follows:

$$\|y - \hat{x}\|_2 + \rho\|A\hat{x}\|_1$$

where ρ is a weighting parameter which decides how much weight is given to the total variation regularization term compared to the error term.

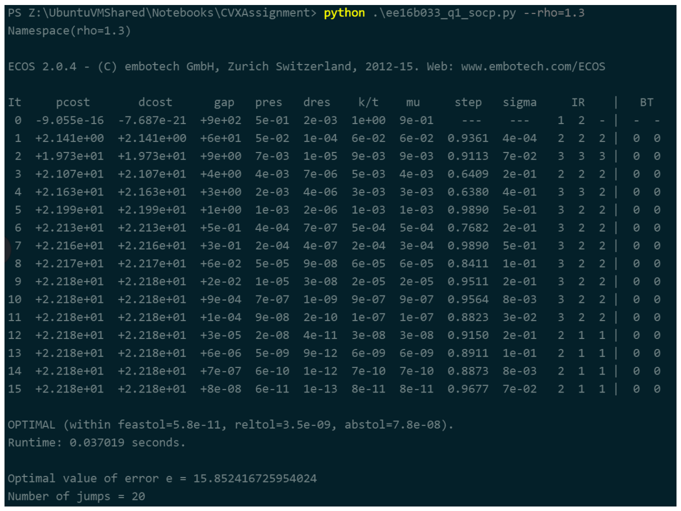
We approximate the number of jumps by the *total variation*, which is the ℓ_1 norm of the first difference of the signal (given by Ax, where A is defined in the problem). Our optimization problem is unconstrained:

$$\underset{x}{\text{minimize}} \quad \|y - \hat{x}\|_2 + \rho\|A\hat{x}\|_1 \tag{1a}$$

We can formulate this as an SOCP by splitting the ℓ_1 norm into each component, and introducing an auxiliary variable u_i for each of the n components, as well as an auxiliary variable t for the error norm term. The final SOCP optimization problem is:

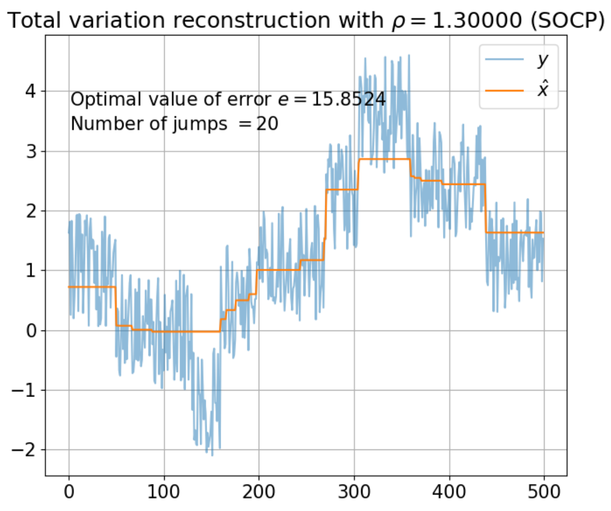
$$\begin{aligned} \underset{x,t,u}{\text{minimize}} \quad & t + \mathbf{1}^T u && \text{(2a)} \\ \text{subject to} \quad & \|y - \hat{x}\|_2 \le t, && \text{(2b)} \\ & -u_i \le (A\hat{x})_i \le u_i \ \forall i = 1, 2, \ldots, n && \text{(2c)} \end{aligned}$$

Jumps in the reconstructed signal are counted by thresholding the absolute value of its first difference. We observe in Fig. 1b, to get 20 jumps in the reconstructed signal, we require a value of $\rho = 1.3$. The error in this case is around 15.85. However, from Fig. 2b, if we set $\rho = 0.6$, the error reduces to around 12.87, but the number of jumps increases to 25. On analyzing the graph of the reconstructed signal, we can see that most of these jumps are actually very small in magnitude, and should ideally not be jumps. This is why a high value of ρ is required to reduce the number of jumps. An animation of the reconstructed signal varying with the value of ρ is also included in the zip file, and is named `socp.gif`. This issue can be tackled by the reweighted ℓ_1 approach, which is briefly described in the next section.



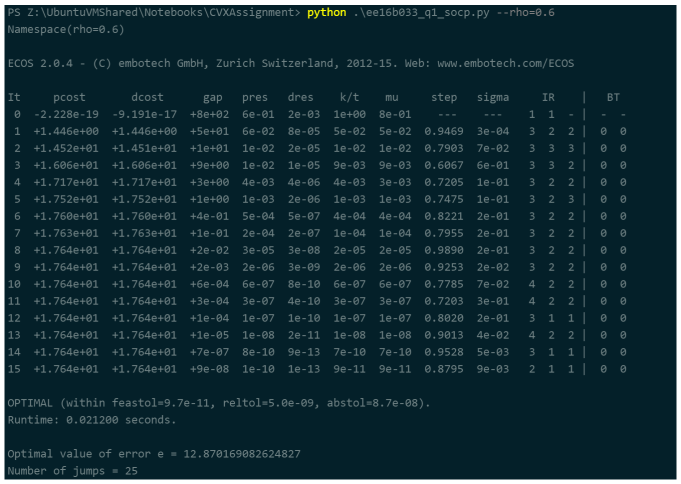
```
PS Z:\UbuntuVMShared\Notebooks\CVXAssignment> python .\ee16b033_q1_socp.py --rho=1.3
Namespace(rho=1.3)

ECOS 2.0.4 - (C) embotech GmbH, Zurich Switzerland, 2012-15. Web: www.embotech.com/ECOS

It     pcost       dcost      gap   pres   dres    k/t    mu     step   sigma     IR    |   BT
 0  -9.055e-16  -7.687e-21  +9e+02  5e-01  2e-03  1e+00  9e-01    ---    ---    1  2  - |  -  -
 1  +2.141e+00  +2.141e+00  +6e+01  5e-02  1e-04  6e-02  6e-02  0.9361  4e-04   2  2  2 |  0  0
 2  +1.973e+01  +1.973e+01  +9e+00  7e-03  1e-05  9e-03  9e-03  0.9113  7e-02   3  3  3 |  0  0
 3  +2.107e+01  +2.107e+01  +4e+00  4e-03  7e-06  5e-03  4e-03  0.6409  2e-01   2  2  2 |  0  0
 4  +2.163e+01  +2.163e+01  +3e+00  2e-03  4e-06  3e-03  3e-03  0.6380  4e-01   3  3  2 |  0  0
 5  +2.199e+01  +2.199e+01  +1e+00  1e-03  2e-06  1e-03  1e-03  0.9890  5e-01   3  2  2 |  0  0
 6  +2.213e+01  +2.213e+01  +5e-01  4e-04  7e-07  5e-04  5e-04  0.7682  2e-01   3  2  2 |  0  0
 7  +2.216e+01  +2.216e+01  +3e-01  2e-04  4e-07  2e-04  3e-04  0.9890  5e-01   3  2  2 |  0  0
 8  +2.217e+01  +2.217e+01  +6e-02  5e-05  9e-08  6e-05  6e-05  0.8411  1e-01   3  2  2 |  0  0
 9  +2.218e+01  +2.218e+01  +2e-02  1e-05  3e-08  2e-05  2e-05  0.9511  2e-01   3  2  2 |  0  0
10  +2.218e+01  +2.218e+01  +9e-04  7e-07  1e-09  9e-07  9e-07  0.9564  8e-03   3  2  2 |  0  0
11  +2.218e+01  +2.218e+01  +1e-04  9e-08  2e-10  1e-07  1e-07  0.8823  3e-02   3  2  2 |  0  0
12  +2.218e+01  +2.218e+01  +3e-05  2e-08  4e-11  3e-08  3e-08  0.9150  2e-01   2  1  1 |  0  0
13  +2.218e+01  +2.218e+01  +6e-06  5e-09  9e-12  6e-09  6e-09  0.8911  1e-01   2  1  1 |  0  0
14  +2.218e+01  +2.218e+01  +7e-07  6e-10  1e-12  7e-10  7e-10  0.8873  8e-03   2  1  1 |  0  0
15  +2.218e+01  +2.218e+01  +8e-08  6e-11  1e-13  8e-11  8e-11  0.9677  7e-02   2  1  1 |  0  0

OPTIMAL (within feastol=5.8e-11, reltol=3.5e-09, abstol=7.8e-08).
Runtime: 0.037019 seconds.

Optimal value of error e = 15.852416725954024
Number of jumps = 20
```

(a) Output of running the script to solve the SOCP.



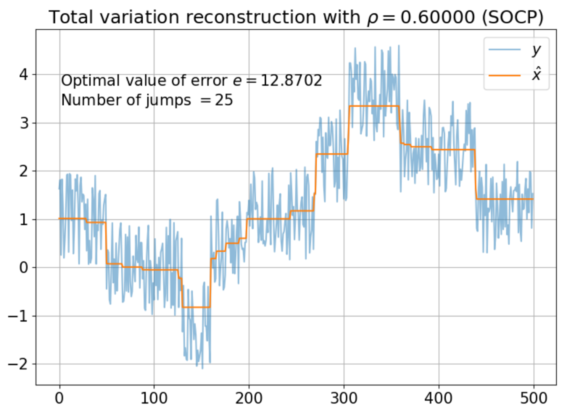
(b) Graph of data and reconstructed signal.

Figure 1: Final signal with 20 jumps. Notice that the error has significantly increased to get fewer jumps.



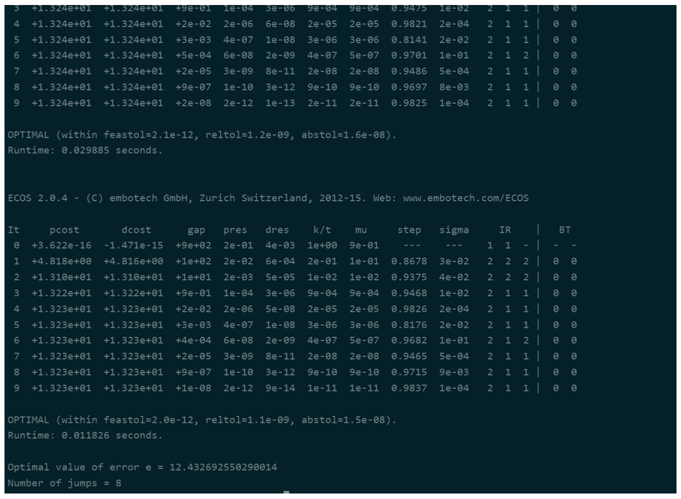
```
PS Z:\UbuntuVMShared\Notebooks\CVXAssignment> python .\ee16b033_q1_socp.py --rho=0.6
Namespace(rho=0.6)

ECOS 2.0.4 - (C) embotech GmbH, Zurich Switzerland, 2012-15. Web: www.embotech.com/ECOS

It     pcost       dcost      gap   pres   dres    k/t    mu     step   sigma     IR    |   BT
 0  -2.228e-19  -9.191e-17  +8e+02  6e-01  2e-03  1e+00  8e-01    ---    ---    1  1  - |  -  -
 1  +1.446e+00  +1.446e+00  +5e+01  6e-02  8e-05  5e-02  5e-02  0.9469  3e-04   3  2  2 |  0  0
 2  +1.452e+01  +1.451e+01  +1e+01  1e-02  2e-05  1e-02  1e-02  0.7903  7e-02   3  3  3 |  0  0
 3  +1.606e+01  +1.606e+01  +9e+00  1e-02  1e-05  9e-03  9e-03  0.6067  6e-01   3  3  2 |  0  0
 4  +1.717e+01  +1.717e+01  +3e+00  4e-03  4e-06  4e-03  3e-03  0.7205  1e-01   3  2  2 |  0  0
 5  +1.752e+01  +1.752e+01  +1e+00  1e-03  2e-06  1e-03  1e-03  0.7475  1e-01   3  2  3 |  0  0
 6  +1.760e+01  +1.760e+01  +4e-01  5e-04  5e-07  4e-04  4e-04  0.8221  2e-01   3  2  2 |  0  0
 7  +1.763e+01  +1.763e+01  +1e-01  2e-04  2e-07  1e-04  1e-04  0.7955  2e-01   3  2  2 |  0  0
 8  +1.764e+01  +1.764e+01  +2e-02  3e-05  3e-08  2e-05  2e-05  0.9890  2e-01   3  2  2 |  0  0
 9  +1.764e+01  +1.764e+01  +2e-03  2e-06  3e-09  2e-06  2e-06  0.9253  2e-02   3  2  2 |  0  0
10  +1.764e+01  +1.764e+01  +6e-04  6e-07  8e-10  6e-07  6e-07  0.7785  7e-02   4  2  2 |  0  0
11  +1.764e+01  +1.764e+01  +3e-04  3e-07  4e-10  3e-07  3e-07  0.7203  3e-01   4  2  2 |  0  0
12  +1.764e+01  +1.764e+01  +1e-04  1e-07  1e-10  1e-07  1e-07  0.8020  2e-01   3  1  1 |  0  0
13  +1.764e+01  +1.764e+01  +1e-05  1e-08  2e-11  1e-08  1e-08  0.9013  4e-02   4  2  2 |  0  0
14  +1.764e+01  +1.764e+01  +7e-07  8e-10  9e-13  7e-10  7e-10  0.9528  5e-03   3  1  1 |  0  0
15  +1.764e+01  +1.764e+01  +9e-08  1e-10  1e-13  9e-11  9e-11  0.8795  9e-03   2  1  1 |  0  0

OPTIMAL (within feastol=9.7e-11, reltol=5.0e-09, abstol=8.7e-08).
Runtime: 0.021200 seconds.

Optimal value of error e = 12.870169082624827
Number of jumps = 25
```

(a) Output of running the script to solve the SOCP.



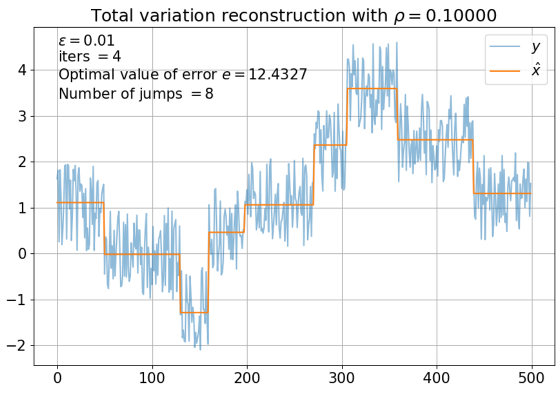
(b) Graph of data and reconstructed signal.

Figure 2: Solving the SOCP for piecewise reconstruction. Final signal with 25 jumps.



```
 3  +1.324e+01  +1.324e+01  +9e-01  1e-04  3e-06  9e-04  9e-04  0.9475  1e-02   2  1  1 |  0  0
 4  +1.324e+01  +1.324e+01  +2e-02  2e-06  6e-08  2e-05  2e-05  0.9821  2e-04   2  1  1 |  0  0
 5  +1.324e+01  +1.324e+01  +3e-03  4e-07  1e-08  3e-06  3e-06  0.8141  2e-02   2  1  1 |  0  0
 6  +1.324e+01  +1.324e+01  +5e-04  6e-08  2e-09  4e-07  5e-07  0.9701  1e-01   2  1  2 |  0  0
 7  +1.324e+01  +1.324e+01  +2e-05  3e-09  8e-11  2e-08  2e-08  0.9486  5e-04   2  1  1 |  0  0
 8  +1.324e+01  +1.324e+01  +9e-07  1e-10  3e-12  9e-10  9e-10  0.9697  8e-03   2  1  1 |  0  0
 9  +1.324e+01  +1.324e+01  +2e-08  2e-12  1e-13  2e-11  2e-11  0.9825  1e-04   2  1  1 |  0  0

OPTIMAL (within feastol=2.1e-12, reltol=1.2e-09, abstol=1.6e-08).
Runtime: 0.029885 seconds.


ECOS 2.0.4 - (C) embotech GmbH, Zurich Switzerland, 2012-15. Web: www.embotech.com/ECOS

It     pcost       dcost      gap   pres   dres    k/t    mu     step   sigma     IR    |   BT
 0  +3.622e-16  -1.471e-15  +9e+02  2e-01  4e-03  1e+00  9e-01    ---    ---    1  1  - |  -  -
 1  +4.818e+00  +4.816e+00  +1e+02  2e-02  6e-04  2e-01  1e-01  0.8678  3e-02   2  2  2 |  0  0
 2  +1.310e+01  +1.310e+01  +1e+01  2e-03  5e-05  1e-02  1e-02  0.9375  4e-02   2  2  2 |  0  0
 3  +1.322e+01  +1.322e+01  +9e-01  1e-04  3e-06  9e-04  9e-04  0.9468  1e-02   2  1  1 |  0  0
 4  +1.323e+01  +1.323e+01  +2e-02  2e-06  5e-08  2e-05  2e-05  0.9826  2e-04   2  1  1 |  0  0
 5  +1.323e+01  +1.323e+01  +3e-03  4e-07  1e-08  3e-06  3e-06  0.8176  2e-02   2  1  1 |  0  0
 6  +1.323e+01  +1.323e+01  +4e-04  6e-08  2e-09  4e-07  5e-07  0.9682  1e-01   2  1  2 |  0  0
 7  +1.323e+01  +1.323e+01  +2e-05  3e-09  8e-11  2e-08  2e-08  0.9465  5e-04   2  1  1 |  0  0
 8  +1.323e+01  +1.323e+01  +9e-07  1e-10  3e-12  9e-10  9e-10  0.9715  9e-03   2  1  1 |  0  0
 9  +1.323e+01  +1.323e+01  +1e-08  2e-12  9e-14  1e-11  1e-11  0.9837  1e-04   2  1  1 |  0  0

OPTIMAL (within feastol=2.0e-12, reltol=1.1e-09, abstol=1.5e-08).
Runtime: 0.011826 seconds.

Optimal value of error e = 12.432692550290014
Number of jumps = 8
```

(a) Output of running the script to solve the reweighted problem.



(b) Notice that only 8 jumps are present.

Figure 3: Reweighted l1 minimization with 4 reweighted iterations. Notice that the error remains relatively low at around 12.43.

2.2 Reweighted ℓ_1 minimization

The code can be found in `ee16b033_q1_rw.py`.

The reweighted ℓ_1 minimization algorithm [1] enhances sparsity in reconstructions by using a majorization-minimization approach to iteratively solve a weighted ℓ_1 minimization problem. It uses the solution of this problem to decide the weights of the next iteration. Briefly,

$$x_{k+1} = \underset{x}{\arg\min} \quad \|W_k \circ x\|_1 \tag{3a}$$

$$W_{k+1} = \frac{1}{|x_{k+1} + \epsilon|}$$

where $\circ$ denotes element-wise multiplication. The details of the algorithm can be found in [1].

Observe from Fig. 3b that with a lower ρ of 0.1, the error remains similar to that in Fig. 2b at around 12.47. The significant difference is in the number and quality of jumps. Only 8 jumps are present, but they are not small in magnitude. An animation of the reconstruction varying with ρ is also included in the zip file, named as `reweighted_l1.gif`. Qualitatively, one can see that this algorithm results in a better piecewise constant reconstruction than just standard ℓ_1 minimization. An animation comparing both methods is also included as `both.gif`.

3 Question 2: Resource Allocation

The code can be found in `ee16b033_q2.py`.

We can initially formulate the optimization problem as:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_j \min\{p_j x_j, p_j q_j + p_j^{disc}(x_j - q_j)\} && \text{(4a)} \\ \text{subject to} \quad & Ax \leq c_{max}, && \text{(4b)} \\ & x \geq 0 && \text{(4c)} \end{aligned}$$

We first convert the maximization to a minimization problem, then introduce auxiliary variables t_j corresponding to each activity.

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & \sum_j t_j && \text{(5a)}\\
\text{subject to} \quad & \max\{-p_j x_j, -p_j q_j - p_j^{disc}(x_j - q_j)\} \leq t_j \ \forall j, && \text{(5b)}\\
& Ax \leq c_{max}, && \text{(5c)}\\
& x \geq 0 && \text{(5d)}
\end{aligned}$$

We can rewrite the max term as two separate constraints to get the final LP formulation:

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & \sum_j t_j && \text{(6a)}\\
\text{subject to} \quad & -p_j x_j \leq t_j \ \forall j, && \text{(6b)}\\
& -p_j q_j - p_j^{disc}(x_j - q_j) \leq t_j \ \forall j, && \text{(6c)}\\
& Ax \leq c_{max}, && \text{(6d)}\\
& x \geq 0 && \text{(6e)}
\end{aligned}$$

On solving, we get the following results (from Fig. 4):

$$x^* = [4, 22.5, 31, 1.5]^T$$

$$r^* = [12, 32.5, 139, 9]^T$$

$$\text{Total revenue} = \sum_j r_j^* = 192.5$$

$$\bar{p} = [3, 1.44444, 4.48387, 6]^T$$

We can make some interpretations based on the solution obtained:

- Activity 3 has the highest price before discount and after discount as well, and also has modest resource consumption, so it is expected to have the highest activity level.
- Activity 1 has a level which is just equal to the discount quantity q_j, so the corresponding constraint would be tight (dual variable = 0).
- Since activities 1 and 4 operate below or at the discount quantity q_j, their average prices are equal to their base prices p_j.
- Since activities 2 and 3 operate well above the discount quantity q_j, their average prices lie between the base and discount prices.

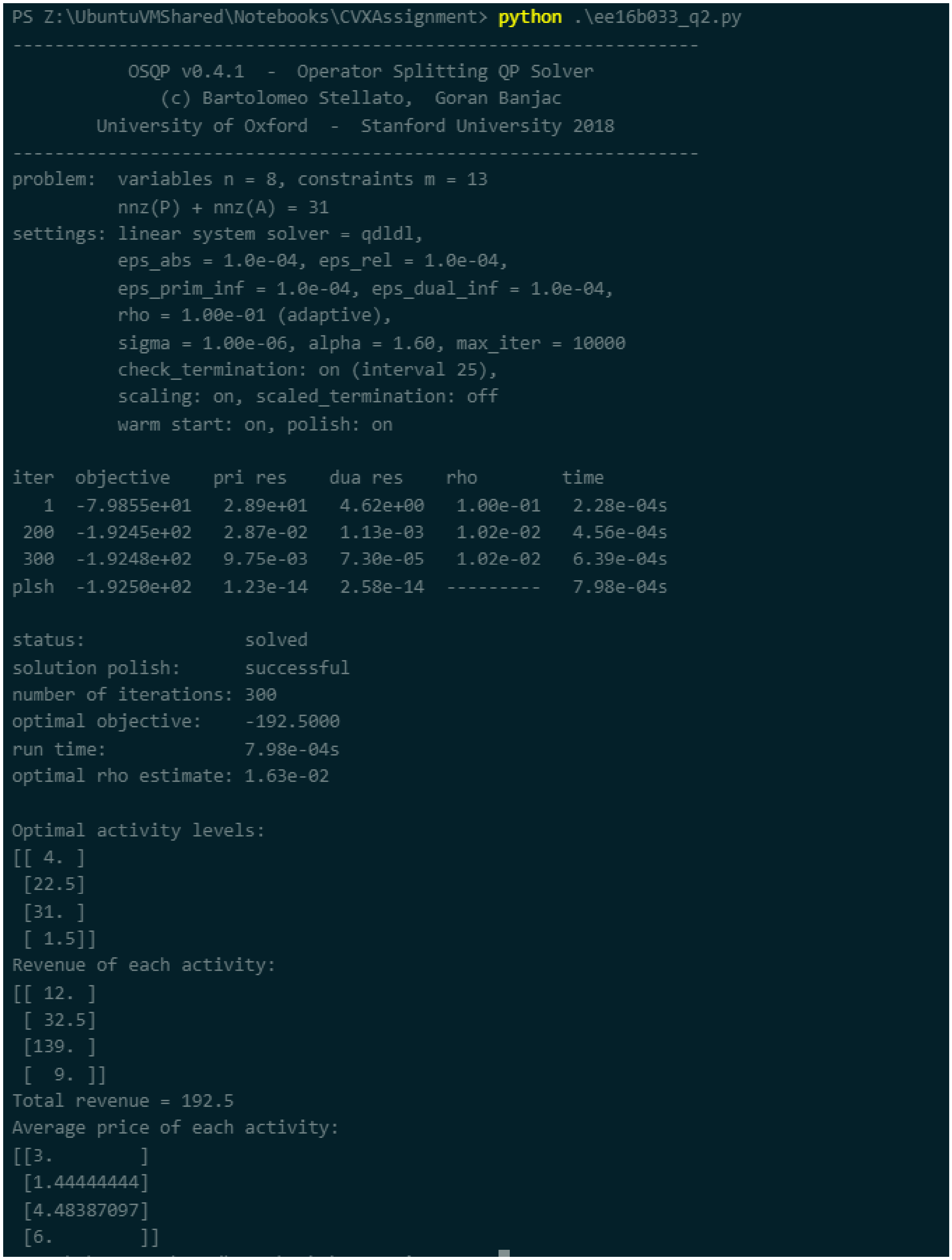
4 Question 3: Matrix Completion

The code can be found in `ee16b033_q3.py`.

Using the epigraph form with auxiliary variable r, we can rewrite the problem in the question as:

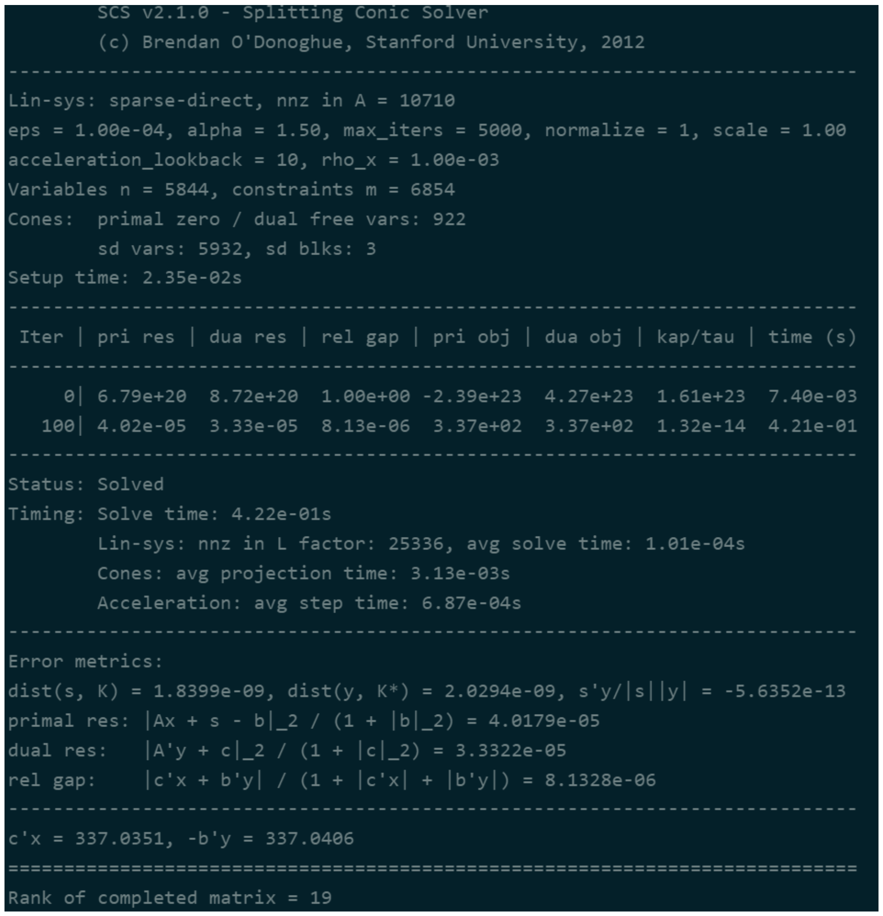
$$\begin{aligned}
\underset{\hat{X}}{\text{minimize}} \quad & r && \text{(7a)}\\
\text{subject to} \quad & \text{rank}(\hat{X}) \leq r, && \text{(7b)}\\
& \hat{X}_{ij} = X_{ij} && \text{(7c)}
\end{aligned}$$

Using the given result, we can rewrite this as



```
PS Z:\UbuntuVMShared\Notebooks\CVXAssignment> python .\ee16b033_q2.py
-----------------------------------------------------------------
           OSQP v0.4.1  -  Operator Splitting QP Solver
              (c) Bartolomeo Stellato,  Goran Banjac
        University of Oxford  -  Stanford University 2018
-----------------------------------------------------------------
problem:  variables n = 8, constraints m = 13
          nnz(P) + nnz(A) = 31
settings: linear system solver = qdldl,
          eps_abs = 1.0e-04, eps_rel = 1.0e-04,
          eps_prim_inf = 1.0e-04, eps_dual_inf = 1.0e-04,
          rho = 1.00e-01 (adaptive),
          sigma = 1.00e-06, alpha = 1.60, max_iter = 10000
          check_termination: on (interval 25),
          scaling: on, scaled_termination: off
          warm start: on, polish: on

iter  objective    pri res    dua res    rho        time
   1  -7.9855e+01   2.89e+01   4.62e+00   1.00e-01   2.28e-04s
 200  -1.9245e+02   2.87e-02   1.13e-03   1.02e-02   4.56e-04s
 300  -1.9248e+02   9.75e-03   7.30e-05   1.02e-02   6.39e-04s
plsh  -1.9250e+02   1.23e-14   2.58e-14   ---------   7.98e-04s

status:               solved
solution polish:      successful
number of iterations: 300
optimal objective:    -192.5000
run time:             7.98e-04s
optimal rho estimate: 1.63e-02

Optimal activity levels:
[[ 4. ]
 [22.5]
 [31. ]
 [ 1.5]]
Revenue of each activity:
[[ 12. ]
 [ 32.5]
 [139. ]
 [  9. ]]
Total revenue = 192.5
Average price of each activity:
[[3.        ]
 [1.44444444]
 [4.48387097]
 [6.        ]]
```

Figure 4: Output of running the script to solve the resource allocation problem (question 2).



```
        SCS v2.1.0 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
----------------------------------------------------------------------------
Lin-sys: sparse-direct, nnz in A = 10710
eps = 1.00e-04, alpha = 1.50, max_iters = 5000, normalize = 1, scale = 1.00
acceleration_lookback = 10, rho_x = 1.00e-03
Variables n = 5844, constraints m = 6854
Cones:  primal zero / dual free vars: 922
        sd vars: 5932, sd blks: 3
Setup time: 2.35e-02s
----------------------------------------------------------------------------
 Iter | pri res | dua res | rel gap | pri obj | dua obj | kap/tau | time (s)
----------------------------------------------------------------------------
     0| 6.79e+20  8.72e+20  1.00e+00 -2.39e+23  4.27e+23  1.61e+23  7.40e-03
   100| 4.02e-05  3.33e-05  8.13e-06  3.37e+02  3.37e+02  1.32e-14  4.21e-01
----------------------------------------------------------------------------
Status: Solved
Timing: Solve time: 4.22e-01s
        Lin-sys: nnz in L factor: 25336, avg solve time: 1.01e-04s
        Cones: avg projection time: 3.13e-03s
        Acceleration: avg step time: 6.87e-04s
----------------------------------------------------------------------------
Error metrics:
dist(s, K) = 1.8399e-09, dist(y, K*) = 2.0294e-09, s'y/|s||y| = -5.6352e-13
primal res: |Ax + s - b|_2 / (1 + |b|_2) = 4.0179e-05
dual res:   |A'y + c|_2 / (1 + |c|_2) = 3.3322e-05
rel gap:    |c'x + b'y| / (1 + |c'x| + |b'y|) = 8.1328e-06
----------------------------------------------------------------------------
c'x = 337.0351, -b'y = 337.0406
============================================================================
Rank of completed matrix = 19
```

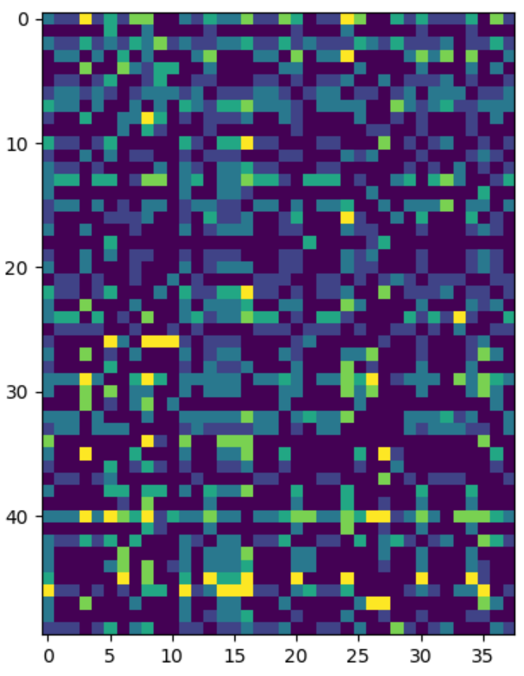
Figure 5: Output of running the script to solve the Netflix problem (question 3).

$$\begin{aligned}
\underset{\hat{X}, Y, Z}{\text{minimize}} \quad & r && \text{(8a)}\\
\text{subject to} \quad & \operatorname{rank}(Y) + \operatorname{rank}(Z) \leq 2r, && \text{(8b)}\\
& \left[\begin{smallmatrix} Y & X \\ X^T & Z \end{smallmatrix}\right] \succeq 0, && \text{(8c)}\\
& \hat{X}_{ij} = X_{ij} && \text{(8d)}
\end{aligned}$$

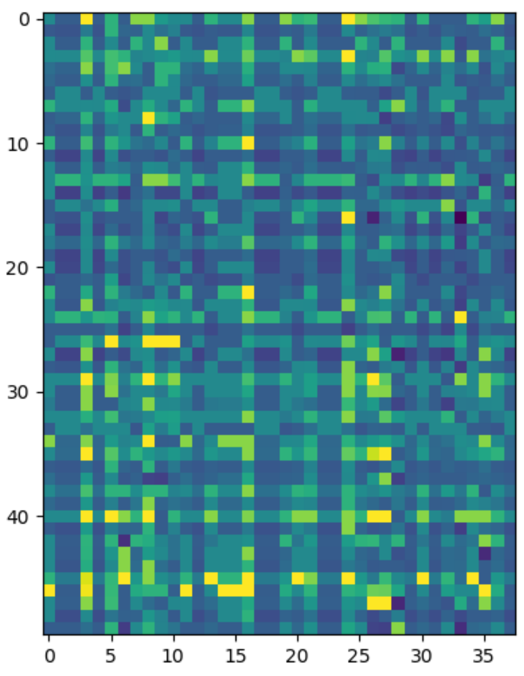
We can now impose positive definiteness of Y and Z to relax the rank into the trace function. We can also go back from the epigraph form to obtain the final SDP formulation as:

$$\begin{aligned}
\underset{\hat{X}, Y, Z}{\text{minimize}} \quad & \mathbf{tr}(Y) + \mathbf{tr}(Z) && \text{(9a)}\\
\text{subject to} \quad & \left[\begin{smallmatrix} Y & X \\ X^T & Z \end{smallmatrix}\right] \succeq 0, && \text{(9b)}\\
& Y \succeq 0, && \text{(9c)}\\
& Z \succeq 0, && \text{(9d)}\\
& \hat{X}_{ij} = X_{ij} && \text{(9e)}
\end{aligned}$$

Note that 9a is affine in the variables Y and Z. The equality constraint 8d is also affine. The remaining constraints are PSD constraints. So, this problem is an SDP. The rank of the completed matrix was found to be **19**. This was estimated by counting the number of singular values of $\hat{X}$ which are greater than some threshold (10^{-6}).



(a) The input data matrix.



(b) The completed matrix.

Figure 6: Comparing the input data and final completed matrices in the Netflix problem (question 3).

References

[1] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.