CS427: Mathematics for Data Science **(Due: 12-Nov-2021)**

Project - 1

Instructor: Dr. Bharath B.N *TAs:* Chandan N., Jayanth S., Shruti M., Sawan S. M

Group 4:

Hariom Meena - 190020017
Ritwik Singh - 180010028
Suryavijoy Kar - 190020039

Contributions:

Problem 1 : Solved by Hariom Meena. LaTeX documentation by Hariom Meena.
Problem 2 : Solved by Ritwik Singh and Suryavijoy Kar. LaTeX documention by Hariom Meena.
Problem 3 and 4 : Discussed among all 3. Various solvers were tested out by all 3 members. Final code written by Ritwik Singh and Suryavijoy Kar. LaTeX documention by Hariom Meena.
Problem 5 : Discussed among all 3. Final code written jointly by Hariom Meena, Ritwik Singh and Suryavijoy Kar. LaTeX documention by Hariom Meena.

Problem 1:

Lets consider the L_0 norm.
L_0 norm is not convex. Let us consider and example to prove it.

$$Let\ x = \begin{pmatrix}1\\0\end{pmatrix}\ and\ y = \begin{pmatrix}0\\1\end{pmatrix}$$

then for $\theta = 1/2 \in (0,1)$ we have,

$$\begin{aligned} ||\theta x + (1-\theta)y||_0 &= 2 \\ but\ \theta||x||_0 + (1-\theta)||y||_0 &= 1 \\ so,\ ||\theta x + (1-\theta)y|| &\geq \theta||x|| + (1-\theta)||y|| \end{aligned}$$

which violates the condition for convex functions. Thus L_0 norm is not a convex function. Consider the given problem:

$$\min_s ||s||_0$$

$$s.t.\ ||y - Cs||_2^2 = 0$$

here, $||s||_0$ is the objective function which is not convex and hence, the given problem is not a convex optimization problem.

Problem 2:

Let $||x||$ be the L_1 norm is all below cases.
We know that

$$||x + y|| \leq ||x|| + ||y|| \qquad (normproperty)$$

and

$$||\alpha x|| = |\alpha|||x||$$
$$||\theta x + (1-\theta)y|| \leq \theta||x|| + (1-\theta)||y||$$
$$since\ (\theta \in (0,1)\ and\ \theta \geq 0).$$

So, $||x||$ or L_1 norm is convex!—-(1)
The equality $||y - Cs||_2^2 = 0$ is equivalent to : $y - Cs = 0$
i.e. $h(s) = y - Cs = 0$ is Affine—-(2)
The given convex problem is

$$\min_s ||s||_1$$
$$s.t.\ ||y - Cs||_2^2 = 0.$$

which is equivalent to

$$\min_s ||s||_1$$
$$s.t.\ y - Cs = 0$$

From (1) and (2) we have : Objective function is convex, equality constraint is affine. So the given problem is convex optimization problem.
Since here the norm is L_1 rather than L_0, its a convex relaxation to the previous problem.
DUAL PROBLEM :
The given problem is as follows:

$$\min_s ||s||_1$$
$$s.t.\ y = Cs$$

Lagrangian for the problem is:

$$L(s, \nu) = ||s||_1 + \nu^T(y - Cs)$$

Lagrangian dual for the problem can be written as :

$$g(\nu) = \inf_s \left\{ ||s||_1 + \nu^T(y - Cs) \right\}$$
$$= \inf_s \left\{ ||s||_1 - (C^T\nu)^T s + \nu^T y \right\}$$

Lets consider the possible cases:
(a) If $C^T\nu \succ 1$ then
we can take $s \to \infty$ and $-(C^T\nu)^T s \to -\infty$,
and $L(s, \nu) \to -\infty$ so, $g(\nu) \to -\infty$
(b) If $C^T\nu \preceq 1$ then

$$||C^T\nu||_\infty = max(allcomponents) \leq 1.$$

Let x,y are vectors

$$x^T y = x_1y_1 + x_2y_2 + ...x_ny_n$$

Now, let max(y_1,y_2,.....y_n) = $||y||_\infty = y_k$ for some k
clearly,

$$\begin{aligned} x^T y &= x_1y_1 + x_2y_2 + ...x_ny_n \\ &\leq x_1y_n + x_2y_n + ...x_ny_n \\ &= (x_1 + x_2 + ...x_n)y_n \end{aligned}$$

Thus,

$$\begin{aligned} x^T y &\leq ||x||_1 y_n \\ x^T y &\leq ||x||_1||y||_\infty \\ also\ y^T x = x^T y &\leq ||x||_1||y||_\infty \end{aligned}$$

Using the result we have:

$$(C^T\nu)^T s \leq ||s||_1||C^T\nu||_\infty \leq ||s||_1$$
$$since\ (||C^T\nu||_\infty \leq 1).$$

So,

$$||s||_1 - (C^T\nu)^T s \geq 0$$

Thus, $\inf\{L(S,\nu)\} = \nu^T y$
i.e. $g(\nu) = \nu^T y$
Hence the dual function is defined as:

$$g(\nu) = \begin{cases} -\infty, & \text{for } (C^T\nu) \succ 1 \\ \nu^T y, & \text{for } (C^T\nu) \preceq 1 \end{cases}$$

So $g(\nu)$ is always concave and we need to solve the following dual problem:

$$\max_s y^t\nu$$
$$s.t.\ ||C^T\nu||_\infty \leq 1$$

Slater's condition is given as :
For the given problem, there must be at least one s^* in domain s.t.

$$y = Cs^*$$

So, then the strong duality condition will hold.
Strong Duality :
We will assume strong duality exists and then find a solution to primal using dual. If the solution turns out to be unique for the primal problem then our assumption is correct.
Lets say strong duality exists, then solution ν^* to dual and s^* to primal satisfies the following:

$$\begin{aligned} y^T\nu^* &= ||s^*||_1 \\ \Rightarrow (Cs^*)^T\nu^* &= ||s^*||_1 \\ \Rightarrow s^{*T}(C^T\nu^*) &= ||s^*||_1 \\ \Rightarrow ||s^*||_1 &= \sum_{i=1}^{n} (C^T\nu^*)(i)s^*(i) \end{aligned}$$

For strong duality, we take equality

$$\Rightarrow (C^T\nu^*)(i) = sign(s^*(i)) - - - -(1)$$

We will now prove that the solution s^* is unique to the primal problem.
Generally for an primal problem optimal value p^* and dual problem optimal value d^* we have:

$$d^* \leq p^*$$

For any feasible s_0^* solution to the primal problem and a particular solution ν^* to dual, we have:

$$y^T\nu^* \leq ||s_0^*||_1$$

$$\begin{aligned}\Rightarrow ||s_0^*||_1 \geq y^T\nu^* &= (Cs^*)^T\nu^* \\ &= s^{*T}(C^T\nu^*) \\ &= \sum_{i=1}^{n} s^*(i)(C^T\nu^*)(i) \\ &= ||s^*||_1 \quad (from(1))\end{aligned}$$

This means s_0^* is unique and equal to s^* which is obtained from dual problem.
So we have:

$$Strong\ Duality \Rightarrow Unique\ Primary\ Solution.$$

Hence, our assumption of strong duality was correct and we got a unique solution. So,

$$duality\ gap = 0$$

Google Drive link for related files Group 4

Problem 3:

Q3-4 Github
The first section of the 'Problem 3 and 4.ipynb' notebook has the solution to problem 3.
The dataset (A_inv.npy, C.npy, y.npy, Incomplete.png) is provided and we have to find a solution to the optimization problem mentioned in Problem 2 using a computer. We use python language to implement the same.
We use 'cvxpy' and 'numpy' packages to implement the code. The constraint of the original problem $||y - Cs||_2^2 = 0$ is convex and not affine, but the 'cvxpy' function demands that the constraints be affine. So we tweak the constraints a bit to make it affine. Instead of $||y - Cs||_2^2 = 0$, we use $y = Cs$. It is quite intuitive that both these conditions represent the same constraint. This solves the problem of constraints not being affine.
We tried various solvers such as 'CPLEX', 'SCS', 'OSQP' and 'CVXOPT'. Except for 'OSQP', all other solvers ran for more than 20 mins without converging and could not solve the problem at all. Of all the solvers we tried, only 'OSQP' was able to solve the problem.
The optimised value of s is exported to local drive as 's.npy'.

Problem 4:

Q3-4 Github
The second section of the 'Problem 3 and 4.ipynb' notebook has the solution to problem 4. In the previous section, we trained our model and we got the optimised value of s, which we stored in 's.npy'. In this section, we have to reconstruct an original complete/uncorrupted image from the dataset provided (A_inv.npy, C.npy, y.npy, Incomplete.png) using the optimised value of s which we obtained in the previous section.
We use 'scipy', 'numpy', 'matplotlib' and 'imageio' packages in our code. To find the value of $x = As$, we tried different techniques. We tried to find A from A_inv, but the matrix A_inv has a size of (9900,9900) which is quite large and no library function is able to properly find the inverse. We also used the Inverse 2D Discrete Cosine Transform to find the value of $x = As$. The Inverse 2D Discrete Cosine Transform gave a much better result than calculating A from A_inv. s has a shape of (9900,1) but the Inverse 2D Discrete Cosine Transform takes input in the form of a 2D matrix, so we reshape it to (99,100). The Inverse 2D Discrete Cosine Transform computes $x = As$ and returns x in the form of a matrix which we store in x_img. Some values in x_img are beyond the accepted

values for images. In order to avoid clipping of such values, we used 'min-max scaler' on x_img to scale all values from 0 to 1.
In the last part of the code, we plotted both the corrupted image and the reconstructed image. It is difficult to infer anything from the corrupted image but the reconstructed image looks like a zoomed image of a car. The reconstructed image is not very sharp but given that the corruption is very high, we can say that the reconstruction is very good.

Problem 5:

Q5 Github
We have taken an rgb image '1.png', which is a picture of a cat. We use the 'create_data_for_assignment.ipynb' notebook to create all the datasets (A_inv.npy, y.npy , C.npy , incomplete.png). Since the image is in rgb format, we treat color components separately and so we get a total of 3 datasets corresponding to each component. We have used 2 seperate values of corruption 50% and 70% and compared the reconstruction. More details regarding the code can be found in the readme file. The procedure to solve the optimised values for each component is exactly similar to what is done in problem 3 and 4.
For 70% corruption, we see that the reconstructed image looks similar to the original image but is very blurry. This is because of the high value of corruption. The corrupted image does not look anything like the original image. So we can say that our reconstruction was pretty good.
For 50% corruption, we see that the reconstructed image looks similar to the original image but is blurry. The corrupted image looks a bit like the original but it is still not understandable. The reconstructed image for 50% corruption looks much clearer and sharper than the reconstructed image for 70% corruption.