E0 230 CMO Assignment 3 - 27/10/2025

Instructions

1. Read all instructions before starting the assignment.
   Read them again before submitting the assignment.
2. Submissions that do not adhere to the instructions will be given **zero marks**.
3. This is an **individual assignment**, all work including code should be your own.
4. **Do not use AI generated code.**
5. Any form of academic dishonesty will be treated as per the IISc and CSA academic integrity policy.
6. You will submit a `PDF` file and a **single** `.py` file for this assignment. Upload the files directly; **do not zip them**.
   (a) Name the `pdf` file **CMO2025A3_vwxyz.pdf** where vwxyz is your five-digit SR number. The pdf file should be generated using the **LaTeX template provided in the Class Materials folder** in the Files Section of the Team. Your pdf should include all values, justifications, and graphs (if any) that you are asked to report.
   (b) Name the `.py` file **CMO2025A3_vwxyz.py** where vwxyz is your five-digit SR number. Your python file should contain all code that you used to solve this assignment. **Only `.py` files are accepted**; no notebooks.
7. We will not open or evaluate any other file submitted.
8. Incomplete submissions (without the `pdf` or `.py` file) will not evaluated.
9. This assignment is due by 23:59, 5th November.
10. Late submissions will incur a 20% penalty per day.

Oracle Instructions

1. Run all code for this assignment in a virtual environment with Python version 3.10.
2. Unzip the Oracle into the directory you are working in.
3. Do not modify the `oracle_2025A3` folder.
4. Place your `.py` file and the `oracle_2025A3` folder in the **same directory**.
5. Import the Oracle using `from oracle_2025A3 import f1`.
6. Call the Oracle:
   - `f1(12345)`
     This will save your dataset in the same folder with the name `data_12345.csv`
7. Set up the Oracle by 23:59, 30th October.
8. No support will be provided for Oracle issues after this deadline.

1 LASSO Regression and KKT Conditions (18 marks)

Consider the optimization problem of a convex objective function with an added sparsity-promoting constraint.

$$\min_{\beta \in \mathbb{R}^n} \; f(\beta) \quad \text{s.t.} \quad \|\beta\|_1 \leq t \tag{1}$$

However, in practice we optimize

$$\min_{\beta \in \mathbb{R}^n} \; f(\beta) + \lambda \|\beta\|_1 \tag{2}$$

where $\lambda > 0$ is known a regularization parameter.

When the objective function is of the form:

$$f(\beta) = \frac{1}{2}\|X\beta - y\|_2^2$$

This becomes the problem of linear regression, but with an added sparsity-promoting regularizer (also known as LASSO regression).

You are provided with a small dataset in `data.csv` containing 50 samples with 15 features where $X \in \mathbb{R}^{50\times 15}$ and response $y \in \mathbb{R}^{50}$.

1. (2 points) Derive the KKT conditions for the optimization problem (1) in terms of the subgradient of $\|\beta\|_1$.
2. (4 points) Is optimizing (1) equivalent to optimizing (2)? Reason out why optimization problem (2) is solved in practice instead of solving (1).

3. (4 points) Implement LASSO regression for the Linear objective function using `CVXPY` for three values of λ: 0.01, 0.1 and 1. Plot the number of fitted (non-zero) coefficients against λ, to compare the sparsity of β^*.

4. (4 points) Verify if the KKT conditions hold for your optimal β^* and report your observations.

5. (4 points) Consider the case where two features are highly correlated. Duplicate one feature column in X and repeat the experiment. What happens to the LASSO solution? Relate your observation to the KKT subgradient condition.

2 Duality and Sparse Feature Selection (17 marks)

Consider the LASSO dual problem. Recall that the LASSO primal can be rewritten as:

$$\min_{\beta} \ \frac{1}{2}\|X\beta - y\|_2^2 + \lambda\|\beta\|_1$$

and its dual form can be shown to be:

$$\max_{u} \ -\frac{1}{2}\|u\|_2^2 + y^\top u \quad \text{s.t.} \quad \|X^\top u\|_\infty \leq \lambda$$

1. (4 points) Derive this dual problem starting from the primal formulation using the Lagrangian. Clearly state your steps and assumptions.

2. (3 points) Show that strong duality holds under mild assumptions on X.

3. (4 points) For the same dataset as Question 1, implement the dual formulation using `CVXPY` for $\lambda = 0.01$, 0.1 and 1.

4. (2 points) Check how closely the optimum value obtained u^* satisfies the relation with β^* obtained in 1.3. Express in terms of L2 norm.

5. (4 points) Explain, in your own words, how the dual constraint $\|X^\top u\|_\infty \leq \lambda$ enforces sparsity in the primal coefficients β^*.

3 Question 3 (15 marks)

1. (5 points) **Projections in a navigation problem.** A robot at position $y \in \mathbb{R}^2$ must stay inside a safe zone. Two possible safe zones are:

   (a) A circular base station $C_1 = \{x : \|x\|_2 \leq 5\}$,

   (b) A rectangular corridor $C_2 = \{x : -3 \leq x_1 \leq 3,\ 0 \leq x_2 \leq 4\}$.

   - Implement functions to compute projections $\Pi_{C_1}(y)$ and $\Pi_{C_2}(y)$.
   - **Submit:** Python code (`PROJ_CIRCLE`, `PROJ_BOX`), and plots showing at least 3 sample robot positions with their projections onto each safe zone.

2. (5 points) **Separating hyperplane in a classification story.** A company has two groups of customers: - Group A: customers who always pay on time, represented as points inside the unit circle $C_A = \{x : \|x\|_2 \leq 1\}$, - Group B: high-risk customers, all lying in the half-space $C_B = \{x : x_1 \geq 3\}$. By the **Separating Hyperplane Theorem**, the company wants to find a hyperplane that separates C_A and C_B.

   - Write code to compute such a separating hyperplane (normal vector and offset).
   - **Submit:** Python code (`SEPARATE_HYPERPLANE`), and a plot showing C_A, C_B, and the separating hyperplane.

3. (5 points) **Farkas lemma in a supply-chain model.** Suppose a factory must meet demand d using resources $x \in \mathbb{R}^2$ subject to capacity constraints $Ax \leq b$. Sometimes, no feasible plan exists. In that case, by **Farkas lemma**, there exists a vector $y \geq 0$ certifying infeasibility. Consider the system:

$$x_1 + x_2 \leq -1, \quad -x_1 \leq 0, \quad -x_2 \leq 0.$$

   - Write code to check feasibility (using CVXPy). If infeasible, compute a Farkas certificate y.
   - **Submit:** Python code (`CHECK_FARKAS`), the certificate y, and a short explanation (3–4 sentences) of what y means in the supply-chain context.

Function Specifications (for coding consistency)

- **Projection onto circle**
  - **Function Name:** `PROJ_CIRCLE`
  - **Inputs:**
    1. `y`: point to project (NumPy array of length 2).
    2. `center`: centre of circle (default = `np.array([0.0,0.0])`).
    3. `radius`: radius of circle (default = 5.0).
  - **Outputs:**
    1. `y_proj`: projection of `y` on the closed Euclidean ball (NumPy array of length 2).
- **Projection onto box**
  - **Function Name:** `PROJ_BOX`
  - **Inputs:**
    1. `y`: point to project (NumPy array of length 2).
    2. `low`: lower corner of box (default = `np.array([-3.0,0.0])`).
    3. `high`: upper corner of box (default = `np.array([3.0,4.0])`).
  - **Outputs:**
    1. `y_proj`: projection of `y` on the box (NumPy array of length 2).

- **Separating hyperplane (geometry / classification)**
  - **Function Name:** `SEPARATE_HYPERPLANE`
  - **Inputs:**
    1. No required input for the canonical instance (unit circle vs half-space). Optionally: callable descriptions of sets C_A and C_B (for advanced students).
  - **Outputs:**
    1. `n`: normal vector of hyperplane (NumPy array of length 2).
    2. `c`: offset (scalar) so that hyperplane is $\{x : n^\top x = c\}$.
    3. `a_closest`, `b_closest`: the closest points in C_A and C_B used to construct the hyperplane.
- **Farkas lemma / infeasibility check**
  - **Function Name:** `CHECK_FARKAS`
  - **Inputs:**
    1. (Optional) `A`, `b` defining inequalities $Ax \leq b$. If omitted, use the provided supply-chain instance.
  - **Outputs:**
    1. `feasible`: boolean flag (True if feasible).
    2. If infeasible: `y_cert` (NumPy array) a Farkas certificate satisfying $y \geq 0$, $A^\top y = 0$ (numerically), and $b^\top y < 0$ (numerically).
    3. Diagnostic info (objective value, solver status).