# IML: Internal 1

Srikanth Pai, Asst. Professor, MSE, Chennai

17th Feb 2026

Total Marks: 34 (24 Written + 10 Programming) | Time: 90 Minutes
*Note: 28 marks available in Section A; score capped at 24. Answer all parts of a question for full credit.*

## Section A: Written Theory

1. **Conceptual Intuition** *(6 Marks)*

   State whether the following are **True** or **False**. Provide a **one-sentence** justification for each.

   **1.1.** $\mathbb{E}[X \mid Y] = \mathbb{E}[X]$ if and only if $X$ and $Y$ are independent.

   **1.2.** If $A$ is a real symmetric matrix, then the solution to $\min_{\|x\|=1} x^\top Ax$ is the eigenvector corresponding to the largest eigenvalue.

   **1.3.** In K-means, the objective $J = \sum_{i=1}^{n} \|x_i - \mu_{c(i)}\|^2$ is guaranteed to be non-increasing after every assignment step and every centroid update step of Lloyd's algorithm.

   **1.4.** For a differentiable convex function $f$, if $\nabla f(x^*) = 0$, then $x^*$ is a global minimum.

   **1.5.** Replacing $w$ with $2w$ and $b$ with $2b$ in a logistic regression model leaves the decision boundary unchanged but changes the predicted probability $\hat{y} = \sigma(w^\top x + b)$ for every point not on the boundary.

   **1.6.** MAP estimation reduces to MLE when the prior over the model parameters is uniform.

2. **Constrained Optimisation & KKT** *(8 Marks)*

   Consider the following optimisation problem:

   $$\min_{x \in \mathbb{R}^2} \frac{1}{2}(x_1^2 + x_2^2) \quad \text{subject to} \quad x_1 + x_2 \geq 2, \quad 2x_2 \geq 1$$

   **2.1.** **(3 Marks)** Write the Lagrangian $L(x_1, x_2, \lambda_1, \lambda_2)$ for this problem. Ensure you convert the constraints to the standard form $g(x) \leq 0$.

**2.2. (3 Marks)** The unconstrained minimum is at $(0, 0)$. For the constrained case, find the saddle point $(x_1^*, x_2^*)$ of $L$ and the corresponding multipliers $(\lambda_1, \lambda_2)$.

**2.3. (2 Marks)** Identify which constraints are **active** at the solution. Use the KKT condition of **complementary slackness** to justify why one multiplier is zero and the other is non-zero.

3. **New Regression** *(8 Marks)*

Assume a dataset $\{(x_i, y_i)\}_{i=1}^N$ where $y_i \in \{0, 1, 2, \dots\}$. We model the response using a Poisson distribution:

$$P(y \mid \lambda) = \frac{e^{-\lambda}\lambda^y}{y!}, \qquad \text{with} \quad \lambda_i = e^{w^\top x_i}$$

**3.1. (4 Marks)** Derive the Negative Log-Likelihood (NLL) and formulate the loss function $J(w)$ by ignoring terms that do not depend on $w$.

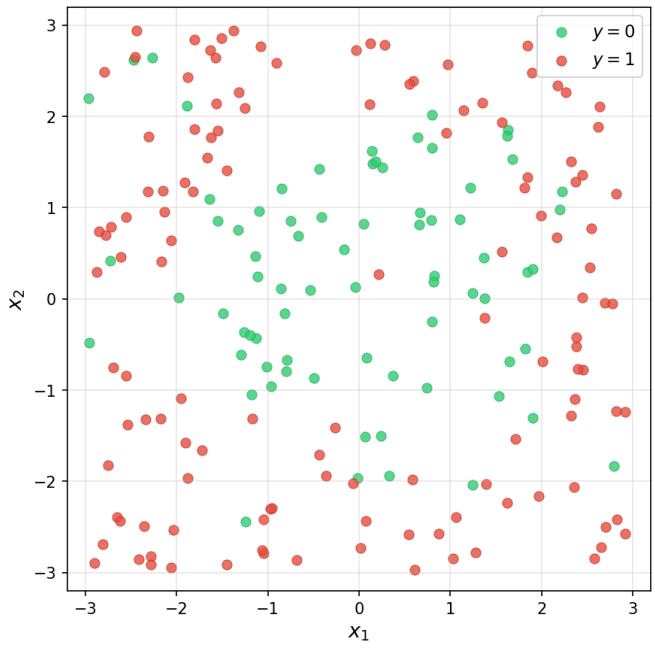
**3.2. (4 Marks)** Compute the gradient $\nabla_w J(w)$ and the Hessian $\nabla_w^2 J(w)$. Use this result to determine if the loss function is convex.

4. **Logistic Regression: Feature Design** *(6 Marks)*

A function $\phi(x_1, x_2)$ is called a *feature* of the data $(x_1, x_2)$ and $\phi$ is called a *feature transformation*. If the feature is a real number instead of a vector, we call it a *scalar feature*. Note that each coordinate is a scalar feature.

For some data sets, it is better to perform regression on features. For instance if $(\phi_1, \phi_2)$ is a feature vector, then the decision boundary of the classifier will look like $w_1\phi_1(x) + w_2\phi_2(x) + b = 0$.

A data scientist can sometimes design a feature so that it works well for the given data set. This is one of those times. The figure below shows a binary classification dataset with $y = 0$ (green) and $y = 1$ (red).




**4.1. (2 Marks)** Propose a scalar feature transformation $\phi(x_1, x_2)$ that allows logistic

regression to classify this data well. Write down the decision boundary in the original $(x_1, x_2)$ space.

**4.2. (4 Marks)** Explain why standard logistic regression on $(x_1, x_2)$ performs poorly here. A student suggests instead fitting logistic regression on the feature vector $(\phi_1, \phi_2) = (x_1^2, x_2^2)$ rather than your scalar $\phi$. Will this work? Is it a better or worse model than yours, and why?

---

# Section B: Programming (10 Marks)

**Task: Manhattan K-Means Step**

Standard K-means uses Euclidean distance to assign points to centroids. In this question you will implement one step of a modified K-means that uses **Manhattan distance** (L1) instead.

The Manhattan distance between two points $a, b \in \mathbb{R}^D$ is:

$$d(a, b) = \sum_{j=1}^{D} |a_j - b_j|$$

One step of the algorithm:

- **Assignment:** Assign each point $x_i$ to the nearest centroid: $c(i) = \arg\min_k d(x_i, \mu_k)$
- **Update:** Recompute each centroid as the mean of its assigned points: $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$

Write a function `kmeans_step(X, centroids)` that performs **one step** of the above and returns the updated centroids.

```
import numpy as np

def kmeans_step(X, centroids):
    """
    Parameters:
        X         : np.ndarray of shape (N, D)
        centroids : np.ndarray of shape (K, D)
    Returns:
        new_centroids : np.ndarray of shape (K, D)
    """
    return
```

## Public Test Cases

| TC | X | centroids | Expected Output |
|---|---|---|---|
| 1 | `[[0],[1],[2],[9],[10],[11]]` | `[[0],[9]]` | `[[1.0],[10.0]]` |
| 2 | `[[0,0],[3,0],[0,3], [10,10],[13,10],[10,13]]` | `[[0,0],[10,10]]` | `[[1.0,1.0], [11.0,11.0]]` |
| 3 | `[[0],[1],[2],[9],[10],[11]]` | `[[1],[10]]` | `[[1.0],[10.0]]` |

*Note: You can try your function at* ***online-python.com*** *before submitting. Private hidden test cases will also be used for final grading.*