Multimodal Model Homework 3

April 2, 2025

2023141460251 孙浩翔

Question 1: 请简述 PCA 与 LDA 的关系

主成分分析（PCA）与线性判别分析（LDA）均为线性降维技术，但其目标与假设不同：

1.1 PCA

- **PCA**：无监督方法，旨在最大化数据的总方差。给定数据矩阵 $X \in \mathbb{R}^{n \times d}$（已中心化），计算协方差矩阵：

$$\Sigma = \frac{1}{n-1} X^\top X \tag{1}$$

通过特征分解 $\Sigma v = \lambda v$ 找到投影方向 v。

1.2 LDA

- **LDA**：有监督方法，旨在最大化类间分离并最小化类内差异。定义类内散度矩阵 S_w 和类间散度矩阵 S_b：

$$S_w = \sum_{i=1}^{C} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^\top, \quad S_b = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^\top \tag{2}$$

优化目标函数：

$$J(w) = \frac{w^\top S_b w}{w^\top S_w w} \tag{3}$$

1.3 关系

- PCA 不依赖标签，适合数据压缩；LDA 利用类别信息，适合分类。
- PCA 基于全局协方差，LDA 基于类间与类内散度的比值。

Question 2: LDA 的目标是什么？

LDA 的目标是通过线性变换 $y = w^\top x$ 将高维数据投影到低维空间，使得：

2.1 目标概述

- 不同类别间的分离最大化（由 S_b 度量）。
- 同一类别内的分散最小化（由 S_w 度量）。

2.2 数学表达

数学上，目标函数为：

$$J(w) = \frac{w^\top S_b w}{w^\top S_w w} \tag{4}$$

其中：

- $S_w = \sum_{i=1}^{C} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^\top$，类内散度矩阵。
- $S_b = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^\top$，类间散度矩阵。
- $\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x$，$\mu = \frac{1}{n} \sum_{i=1}^{C} N_i \mu_i$。

Question 3: 请简述二类 LDA 的推导过程

二类 LDA（类别 C_1 和 C_2）的推导如下：

3.1 定义散度矩阵

1.
$$S_w = \sum_{x \in C_1} (x - \mu_1)(x - \mu_1)^\top + \sum_{x \in C_2} (x - \mu_2)(x - \mu_2)^\top \tag{5}$$
$$S_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top \tag{6}$$

3.2 目标函数

2.
$$J(w) = \frac{w^\top S_b w}{w^\top S_w w} = \frac{[w^\top (\mu_1 - \mu_2)]^2}{w^\top S_w w} \tag{7}$$

3.3 优化

3. 求解广义特征值问题：
$$S_b w = \lambda S_w w \tag{8}$$
   因为 S_b 秩为 1，解为：
$$w = S_w^{-1} (\mu_1 - \mu_2) \tag{9}$$

3.4 结果

4. 投影数据 $y = w^\top x$。

Question 4: 请简述 LDA 算法步骤，并写出伪代码

4.1 算法步骤

1. 计算类别均值：
$$\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x \tag{10}$$

2. 计算全局均值：
$$\mu = \frac{1}{n} \sum_{i=1}^{C} N_i \mu_i \tag{11}$$

3. 计算散度矩阵：
$$S_w = \sum_{i=1}^{C} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^\top, \quad S_b = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^\top \tag{12}$$

4. 求解特征值问题：
$$S_w^{-1} S_b w = \lambda w \tag{13}$$
取前 k 个特征向量组成 W。

5. 投影数据：
$$Y = XW \tag{14}$$

4.2 伪代码

```
import numpy as np

def lda(X, y, k):
    # 初始化
    n, d = X.shape
    classes = np.unique(y)
    C = len(classes)

    # 计算均值
    mu = np.mean(X, axis=0)
    mu_i = [np.mean(X[y == c], axis=0) for c in classes]
    N_i = [np.sum(y == c) for c in classes]

    # 计算散度矩阵
    S_w = np.zeros((d, d))
```

```python
    for i, c in enumerate(classes):
        X_c = X[y == c]
        diff = X_c - mu_i[i]
        S_w += diff.T @ diff

    S_b = np.zeros((d, d))
    for i, c in enumerate(classes):
        diff = (mu_i[i] - mu).reshape(-1, 1)
        S_b += N_i[i] * (diff @ diff.T)

    # 特征分解
    eigvals, eigvecs = np.linalg.eigh(np.linalg.inv(S_w) @ S_b)
    idx = np.argsort(eigvals)[::-1]
    W = eigvecs[:, idx[:k]]

    # 投影
    Y = X @ W
    return W, Y

# 示例
X = np.random.rand(100, 5)
y = np.random.randint(0, 3, 100)
W, Y = lda(X, y, 2)
```

Question 5: 请简述 GMA 与 CCA 的关系

5.1 GMA

- **GMA（广义多变量分析）**：包含 LDA、PCA、CCA 等方法的框架，通过线性变换提取数据结构。

5.2 CCA

- **CCA（典型相关分析）**：分析两组变量 X 和 Y 的相关性，优化：

$$\rho = \frac{w_x^\top \Sigma_{xy} w_y}{\sqrt{w_x^\top \Sigma_{xx} w_x}\sqrt{w_y^\top \Sigma_{yy} w_y}} \tag{15}$$

5.3 关系

- LDA 是 GMA 的子方法，优化分类；CCA 是 GMA 的子方法，优化相关性。
- LDA 和 CCA 均通过广义特征值问题求解。

Question 6: 请写出 A_i 与 B_i 的表达式

基于 LDA：

6.1 A_i 的表达式

- A_i：第 i 类均值向量：

$$A_i = \mu_i = \frac{1}{N_i} \sum_{x \in C_i} x \tag{16}$$

6.2 B_i 的表达式

- B_i：第 i 类对类间散度的贡献：

$$B_i = N_i(\mu_i - \mu)(\mu_i - \mu)^\top, \quad \mu = \frac{1}{n} \sum_{i=1}^{C} N_i \mu_i \tag{17}$$

Question 7: GMA 算法伪代码

GMA 是广义框架，以下以 LDA 为默认方法：

7.1 伪代码

```
import numpy as np

def gma(X, y=None, k=2, method='lda'):
    if method == 'lda':
        # LDA 实现
        n, d = X.shape
        classes = np.unique(y)
        mu = np.mean(X, axis=0)
        mu_i = [np.mean(X[y == c], axis=0) for c in classes]
        N_i = [np.sum(y == c) for c in classes]

        S_w = np.zeros((d, d))
        for i, c in enumerate(classes):
            X_c = X[y == c]
            diff = X_c - mu_i[i]
            S_w += diff.T @ diff

        S_b = np.zeros((d, d))
        for i, c in enumerate(classes):
            diff = (mu_i[i] - mu).reshape(-1, 1)
            S_b += N_i[i] * (diff @ diff.T)
```

```
        eigvals, eigvecs = np.linalg.eigh(np.linalg.inv(S_w) @ S_b)
        idx = np.argsort(eigvals)[::-1]
        W = eigvecs[:, idx[:k]]

        Y = X @ W
        return W, Y
    return None, None

# 示例
X = np.random.rand(100, 5)
y = np.random.randint(0, 3, 100)
W, Y = gma(X, y, 2, 'lda')
```