多模态学习导论 作业3

> 黄家曦 2022141460084

1.MvDA的主要思想是什么？

通过将不同模态的数据映射到统一的公共空间，并使得数据不同类别之间的散度矩阵尽可能大，同一类别之间的散度矩阵尽可能小，从而对不同视图的数据进行分类和识别。

2.MvDA的目标是什么？

对$X^{(j)}=\{x_{ijk}|i=1,\cdots,c;j=1,\cdots,v;k=1,\cdots,n_{ij}\}$为来自第$j$个试图的样本集，其中$x_{ijk}$是第$j$个视图的第$i$个类的第$k$个样本，$c$为类别数量，$v$为模态数量，$n_{ij}$是第$j$个模态第$i$类的样本数。对于映射向量$W_1-W_n$，样本空间$y=\{y_{ijk}=w_j^Tx_{ijk}|i=1,\cdots,c;j=1,\cdots,v;k=1,\cdots,n_{ij}\}$，MvDA的目标是：

$$(W_1^*,W_1^*,\cdots,W_v^*)=\underset{W_1,W_2,\cdots,W_v}{\arg\max}\frac{Tr(S_B^y)}{Tr(S_W^y)}$$

其中，$Tr(S_B^y)$为类内散布矩阵，$Tr(S_W^y)$为类间散布矩阵。

$$Tr(S_B^y)=\sum_{i=1}^{c}n_i(\mu_i-\mu)(\mu_i-\mu)^T$$
$$Tr(S_W^y)=\sum_{i=1}^{c}\sum_{j=1}^{v}\sum_{k=1}^{n_{ij}}(y_{ijk}-\mu_i)(y_{ijk}-\mu_i)^T$$

通过将S_B^y、S_W^y化简得：

$$S_B^y=W^TDW$$
$$S_W^y=W^TSW$$

其中$D_{jr}=\sum_{i=1}^{c}\frac{n_ijn_ir}{n_i}\mu_{ij}^{(x)}\mu_{ir}^{(x)T}-\frac{1}{n}(\sum_{i=1}^{c}n_{ij}\mu_{ij}^x)(\sum_{i=1}^{c}n_{ir}\mu_{ir}^x)^T$

$$S_{jr} = \begin{cases} \sum_{i=1}^{c} \left(\sum_{k=1}^{n_{ij}} x_{ijk} x_{ijk}^T - \frac{n_{ij} n_{ir}}{n_i} \mu_{ij}^{(x)} (\mu_{ir}^{(x)})^T \right), & j = r \\ -\frac{n_{ij} n_{ir}}{n_i} \mu_{ij}^{(x)} (\mu_{ir}^{(x)})^T, & \text{otherwise} \end{cases}$$

3.MvDA和LDA的联系

MvDA（Multi-view Discriminant Analysis，多视图判别分析）和 LDA（Linear Discriminant Analysis，线性判别分析）因为 MvDA 可以被视为 LDA 的一种扩展，特别是在处理多视图数据方面。
LDA 和 MvDA 的核心思想都是通过最大化类间散度（between-class scatter）和最小化类内散度（within-class scatter)，找到一个能够更好区分不同类别的投影子空间。

- **LDA**：针对单视图数据，目标是找到一个线性投影方向，使得在投影后的低维空间中，同类样本尽可能聚集，不同类样本尽可能分离。
- **MvDA**：继承了 LDA 的思想，但将其扩展到多视图场景，旨在将多个视图的数据映射到一个共享的低维子空间中，同时保持类别间的可分性和视图间的一致性。

LDA 的目标是最优化以下准则：

$$J(W) = \frac{\mathrm{tr}(W^T S_B W)}{\mathrm{tr}(W^T S_W W)}$$

其中：

- S_B 是类间散度矩阵，表示不同类别样本之间的差异。
- S_W 是类内散度矩阵，表示同一类别样本之间的差异。
- W 是投影矩阵。

通过最大化 $J(W)$，可以找到最优的投影方向。

MvDA 将 LDA 的思想扩展到多视图数据。它的目标函数类似于 LDA，但需要综合考虑所有视图的类间散度和类内散度。具体来说，MvDA 定义了一个全局的类间散度矩阵 S_B^{global} 和类内散度矩阵 S_W^{global}，分别整合了所有视图的信息。目标函数为：

$$J(W) = \frac{\mathrm{tr}(W^T S_B^{\text{global}} W)}{\mathrm{tr}(W^T S_W^{\text{global}} W)}$$

MvDA 的关键在于如何构建 S_B^{global} 和 S_W^{global}，使其能够反映多视图数据的特性。
两者的区别主要体现在数据处理方式和应用场景上，LDA 更适合单视图数据，而 MvDA 则专为多视图数据设计，能够更好地利用多模态信息。

4.MvDA通过什么消除的视图/模态之间的差异？

共享子空间学习

MvDA 的关键在于找到一个共享的低维子空间，使得所有视图的数据在这个子空间中具有一致的类别结构。为了实现这一点：

- 不同视图的特征数据被投影到同一个共享子空间。
- 在这个共享子空间中，MvDA 通过优化目标函数，确保不同视图的同类样本尽可能靠近，从而消除视图间的差异。

这种共享子空间的学习过程本质上是对多视图数据进行对齐（alignment），减少由于视图特性不同而导致的分布差异。

联合优化框架

MvDA 使用了一个联合优化框架，同时考虑了所有视图的信息。具体来说：

- 它定义了一个全局的类间散度矩阵 S_B^{global} 和类内散度矩阵 S_W^{global}，分别综合了所有视图的类间和类内信息。
- 通过最大化 S_B^{global} 和最小化 S_W^{global}，MvDA 确保不同视图的同类样本在共享子空间中具有相似的分布。

这种方法通过全局优化的方式，自然地消除了视图之间的差异。

跨视图一致性约束

MvDA 引入了跨视图一致性（cross-view consistency）的约束条件，以进一步减少视图间的差异。
具体来说：

- 对于同一类别的样本，MvDA 要求不同视图的投影结果在共享子空间中尽可能接近。
- 这种约束可以通过正则化项或特定的损失函数实现。例如，可以引入一个惩罚项，衡量不同视图同类样本之间的距离，并将其加入目标函数中。
  通过这种方式，MvDA 强制不同视图的同类样本在共享子空间中对齐，从而消除视图间的差异。

5.MvDA与GMA的不同点是什么？

尽管MvDA和GMA在处理多视图数据并提高分类性能，但在目标、方法和应用场景上存在显著差异。以下是它们的主要不同点：

核心思想

MvDA：

- MvDA 的核心思想是通过最大化类间散度（between-class scatter）和最小化类内散度（within-class scatter），找到一个共享的低维子空间。
- 它特别强调跨视图一致性，确保不同视图的同类样本在共享子空间中尽可能靠近。

GMA：

- GMA 的核心思想是通过联合优化多个视图的特征表示，找到一个能够同时反映所有视图信息的共享子空间。
- 它不仅关注类别间的可分性，还注重视图之间的相关性和互补性。

目标函数

MvDA：

- MvDA 的目标函数基于 Fisher 判别准则，形式为：

$$J(W) = \frac{\mathrm{tr}(W^T S_B^{\text{global}} W)}{\mathrm{tr}(W^T S_W^{\text{global}} W)}$$

  其中 S_B^{global} 和 S_W^{global} 分别是全局类间散度矩阵和全局类内散度矩阵。

GMA：

- GMA 的目标函数通常结合了协方差矩阵分解和正则化项，形式为：

$$J(W) = \mathrm{tr}(W^T X^T L X W) + \lambda \|W\|^2$$

  其中 L 是拉普拉斯矩阵，用于衡量视图间的相关性，λ 是正则化参数。

视图间的关系建模

MvDA：

- MvDA 强调跨视图一致性，通过正则化项或约束条件，强制不同视图的同类样本在共享子空间中对齐。
- 这种一致性约束直接作用于投影后的特征表示，减少了视图间的分布差异。

GMA：

- GMA 更关注视图之间的相关性和互补性，通过构建视图间的关联矩阵（如协方差矩阵或拉普拉斯矩阵），融合多个视图的信息。
- 它不直接强制视图一致性，而是通过优化视图间的相关性来实现信息融合。

6.视图一致性是什么，为什么其有效？

视图一致性是说由于多个视图实际上对应于同一个对象，因此多个视图之间应该存在一些对应关系。
即对两个对应视图变换w_1和w_2,有$w_1 = Rw_2$，根据定理，第i个视图的变换w_1可以等效地表述如下：

$$w_i = X_i\beta_i$$

其中β_i捕捉每个w_i的结构，然后有：

$$X_1\beta_1 = RX_2\beta_2 = X_1\beta_2$$

证明了$\beta_1 = \beta_2$。换言之，对于不同视图，通过β_i捕获到的每个转换w_i的结构相同。
如果v个视图的X_1、X_2、$\cdots$、X_v对应相同的底层对象，它们之间视图变换的结构也是相似的，即β_1、β_2、$\cdots$、β_v应该相互相似，这种相似性就是**视图一致性**，由下式建模：

$$\sum_{i,j=1}^{v} \|\beta_i - \beta_j\|_2^2$$

添加到MvDA的目标函数的分母中来最小化该式子，重新得到的目标函数（MvDA-VC）为：

$$(W_1^*, W_2^*, \cdots, W_v^*) = \underset{W_1,W_2,\cdots,W_v}{\arg\max} \frac{Tr(W^TDW)}{Tr(W^TSW) + \lambda\sum_{i,j=1}^{v}\|\beta_i - \beta_j\|_2^2}$$

其中λ为平衡因子

而视图一致性之所以有效，是因为它能够充分利用多视图数据的互补性和冗余性，同时减少视图间的噪声和不一致性。

7.试写出MvDA算法的伪代码

```
function W = mvda(X, Y, d)
    % MvDA: Multi-view Discriminant Analysis
    % Inputs:
    %   X: Cell array of data matrices, where X{v} is the data matrix of view v.
    %   Y: Class labels (n x 1 vector).
    %   d: Desired dimensionality of the projection space.
    % Output:
    %   W: Projection matrix (shared subspace).

    % Step 1: Initialize parameters
    V = length(X); % Number of views
    n = size(X{1}, 1); % Number of samples
    c = length(unique(Y)); % Number of classes

    % Global mean and class means
    mu_global = zeros(size(X{1}, 2), V); % Global mean for each view
    mu_class = cell(c, V); % Class means for each view
    for v = 1:V
        mu_global(:, v) = mean(X{v}, 1)'; % Global mean of view v
        for i = 1:c
            idx = find(Y == i); % Indices of class i
            mu_class{i, v} = mean(X{v}(idx, :), 1)'; % Class mean of view v
        end
    end

    % Step 2: Construct scatter matrices
    S_B = zeros(size(X{1}, 2) * V); % Between-class scatter matrix
    S_W = zeros(size(X{1}, 2) * V); % Within-class scatter matrix

    for v = 1:V
        for r = 1:V
            for i = 1:c
                idx = find(Y == i); % Indices of class i
                n_i = length(idx); % Number of samples in class i

                % Between-class scatter
                if v == r
                    S_B((v-1)*size(X{1}, 2)+1:v*size(X{1}, 2), ...
                        (r-1)*size(X{1}, 2)+1:r*size(X{1}, 2)) = ...
                        S_B((v-1)*size(X{1}, 2)+1:v*size(X{1}, 2), ...
                            (r-1)*size(X{1}, 2)+1:r*size(X{1}, 2)) + ...
```

```
                n_i * (mu_class{i, v} - mu_global(:, v)) * ...
                (mu_class{i, r} - mu_global(:, r))';
            else
                S_B((v-1)*size(X{1}, 2)+1:v*size(X{1}, 2), ...
                    (r-1)*size(X{1}, 2)+1:r*size(X{1}, 2)) = ...
                    S_B((v-1)*size(X{1}, 2)+1:v*size(X{1}, 2), ...
                        (r-1)*size(X{1}, 2)+1:r*size(X{1}, 2)) + ...
                    -n_i * mu_class{i, v} * mu_class{i, r}';
            end

            % Within-class scatter
            for k = 1:n_i
                x_ik = X{v}(idx(k), :)'; % Sample k in class i of view v
                S_W((v-1)*size(X{1}, 2)+1:v*size(X{1}, 2), ...
                    (r-1)*size(X{1}, 2)+1:r*size(X{1}, 2)) = ...
                    S_W((v-1)*size(X{1}, 2)+1:v*size(X{1}, 2), ...
                        (r-1)*size(X{1}, 2)+1:r*size(X{1}, 2)) + ...
                    (x_ik - mu_class{i, v}) * (x_ik - mu_class{i, r})';
            end
        end
    end
end

% Step 3: Solve generalized eigenvalue problem
[eigVec, eigVal] = eig(S_B, S_W);
[~, idx] = sort(diag(eigVal), 'descend'); % Sort eigenvalues in descending order
W = eigVec(:, idx(1:d)); % Select top d eigenvectors

% Reshape W into a cell array for each view
W = mat2cell(W, size(X{1}, 2) * ones(1, V), d);
end
```