模式识别与计算机视觉：第二次作业

人工智能学院 221300066 季千焜

2025 年 4 月 7 日

1 习题一

(a)

题目中引入了 γ_{ij}, 其标识着样本 $\boldsymbol{x}_j$ 是否被分到了第 i 组, 同时引入了 $\boldsymbol{\mu}_i$ 作为第 i 组的代表, 因此我们使用 γ_{ij} 和 $\boldsymbol{\mu}_i$ 对 $K-means$ 的目标进行形式化.

$K-means$ 的目标是每组的样本彼此相似, 即属于相同组的一对样本之间的距离很小. 为了形式化地定义 $K-means$ 的目标, 我们认为 $\boldsymbol{\mu}_i$ 表示了聚类的中心, 我们的目标是找到数据点分别属于的聚类, 以及一组 '向量 $\{\boldsymbol{\mu}_i\}$, 使得每个数据点和它最近的向量 $\boldsymbol{\mu}_i$ 之间的距离的平方和最小. 因此, 我们就可以定义一个目标函数:

$$J=\sum_{j=1}^{M}\sum_{i=1}^{K}\gamma_{ij}\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2$$

其表示每个数据点和它被分配到的向量 $\boldsymbol{\mu}_i$ 之间距离的平方和, 只需要最小化目标函数 J 即可求解出最优 γ_{ij} 和 $\boldsymbol{\mu}_i$, 进而完成样本的聚类, 即有:

$$\underset{\gamma_{ij},\boldsymbol{\mu}_i}{argmin}\sum_{i=1}^{K}\sum_{j=1}^{M}\gamma_{ij}\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2$$

(b)

第 i 步:

在固定 $\boldsymbol{\mu}_i$ 的情况下, 每个样本 $\boldsymbol{x}_j$ 对应的 J 的每一个贡献成分:

$$J_j=\sum_{i=1}^{K}\gamma_{ij}\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2$$

互相独立, 因此只要最优化每一个 J_j 即可最优化最终的 J.

由于 $\sum_{i=1}^{K}\gamma_{ij}=1$ 且 $\gamma_{ij}\in\{0,1\}$, 因此 $\{\gamma_{ij},i=1,2,\cdots,K\}$ 中只有一个元素为 1, 其他元素均为零. 则易知最优化问题

$$\underset{i}{argmin}\sum_{i=1}^{K}\gamma_{ij}\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2$$

与最优化问题

$$\underset{i}{argmin}\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2$$

等价。

因此我们可以简单地将数据点对应的聚类设置为最近的聚类中心, 形式化地表达即为:

$$\gamma_{ij}=\begin{cases}1, & \text{if } i=\underset{i}{argmin}\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2\\ 0, & \text{otherwise}\end{cases}$$

第 ii 步:

在固定 γ_{ij} 的情况下, J 是一个关于 $\boldsymbol{\mu}_i$ 的二次函数, 为了最小化 J, 我们只需令 J 关于 $\boldsymbol{\mu}_i$ 的导数等于零, 即可有最小值, 即:

$$\frac{\partial J}{\partial\boldsymbol{\mu}_i}=2\sum_{j=1}^{M}\gamma_{ij}(\boldsymbol{x}_j-\boldsymbol{\mu}_i)=0$$

解出 $\boldsymbol{\mu}_i$ 即可得这一步的更新规则, 即: $\boldsymbol{\mu}_i=\frac{\sum_{j=1}^{M}\gamma_{ij}\boldsymbol{x}_j}{\sum_{j=1}^{M}\gamma_{ij}}$

(c)

对于第 i 步:

由于我们有更新后的 γ'_{ij}:

$$\gamma'_{ij}=\begin{cases}1, & \text{if } i=\arg\min_i\left\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\right\|^2\\ 0, & \text{otherwise}\end{cases}$$

因此新目标函数值减去原目标函数值为

$$
\begin{aligned}
J' - J &= \sum_{i=1}^{K}\sum_{j=1}^{M} \gamma'_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2 - \sum_{i=1}^{K}\sum_{j=1}^{M} \gamma_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2 \\
&= \sum_{i=1}^{K}\sum_{j=1}^{M} (\gamma'_{ij} - \gamma_{ij}) \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2 \\
&= \sum_{i=1}^{K} (\left\| \boldsymbol{x}_j - \boldsymbol{\mu}'_i \right\|^2 - \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2) \\
&\leq 0
\end{aligned}
$$

其中 $\boldsymbol{\mu}'_i$ 是离 $\boldsymbol{x}_j$ 最近的簇, 而 $\boldsymbol{\mu}_i$ 可能是任意一个簇, 因此我们有 $J' \leq J$. 即第 i 步会使目标函数 J 的值减小或持平.

对于第 ii 步:

对于任意一个簇 i 来说, 它的簇中心原来是 $\boldsymbol{\mu}_i$, 可能是任意一个向量, 之后被优化为

$$
\boldsymbol{\mu}'_i = \frac{\sum_{j=1}^{K} \gamma_{ij} \boldsymbol{x}_j}{\sum_{j=1}^{K} \gamma_{ij}}
$$

原来的目标函数可以改写为

$$
J = \sum_{j=1}^{M}\sum_{i=1}^{K} \gamma_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2
$$

即交换求和符号, 这样我们只需要证明 $J_j = \sum_{i=1}^{K} \gamma_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2$ 降低或不增加即可.

$$
\begin{aligned}
J'_j - J_j &= \sum_{i=1}^{K} \gamma_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}'_i \right\|^2 - \sum_{i=1}^{K} \gamma_{ij} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2 \\
&= \sum_{i=1}^{K} \gamma_{ij} (\left\| \boldsymbol{x}_j - \boldsymbol{\mu}'_i \right\|^2 - \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|^2) \\
&= \sum_{i=1}^{K} \gamma_{ij} (\boldsymbol{x}_j - \boldsymbol{\mu}'_i + \boldsymbol{x}_j - \boldsymbol{\mu}_i)^{\mathrm{T}} (\boldsymbol{x}_j - \boldsymbol{\mu}'_i - \boldsymbol{x}_j + \boldsymbol{\mu}_i) \\
&= \sum_{i=1}^{K} \gamma_{ij} (\boldsymbol{\mu}_i - \boldsymbol{\mu}'_i)^{\mathrm{T}} (2\boldsymbol{x}_j - \boldsymbol{\mu}'_i - \boldsymbol{\mu}_i) \\
&= (\boldsymbol{\mu}_i - \boldsymbol{\mu}'_i)^{\mathrm{T}} (2(\sum_{i=1}^{K} \gamma_{ij} \boldsymbol{x}_j) - (\sum_{i=1}^{K} \gamma_{ij})(\boldsymbol{\mu}'_i + \boldsymbol{\mu}_i)) \\
&= (\sum_{i=1}^{K} \gamma_{ij}) (\boldsymbol{\mu}_i - \boldsymbol{\mu}'_i)^{\mathrm{T}} (2 \frac{\sum_{i=1}^{K} \gamma_{ij} \boldsymbol{x}_j}{\sum_{i=1}^{K} \gamma_{ij}} - \boldsymbol{\mu}'_i - \boldsymbol{\mu}_i) \\
&= (\sum_{i=1}^{K} \gamma_{ij}) (\boldsymbol{\mu}_i - \boldsymbol{\mu}'_i)^{\mathrm{T}} (2\boldsymbol{\mu}'_i - \boldsymbol{\mu}'_i - \boldsymbol{\mu}_i) \\
&= -(\sum_{i=1}^{K} \gamma_{ij}) (\boldsymbol{\mu}_i - \boldsymbol{\mu}'_i)^{\mathrm{T}} (\boldsymbol{\mu}_i - \boldsymbol{\mu}'_i) \\
&\leq 0
\end{aligned}
$$

因此我们有 $J'_j - J_j$, 也即有第 ii 步会使目标函数 J 的值降低或不增加.

下面证明 Lloyd 算法会停止:

假设算法不会在有限步内停止, 则目标函数的值 J 一直在变化. 由 (1) 可知, 目标函数 J 的值降低或不增加, 又因为 J 一直在变化, 可以将一系列 J 的值视作严格单调递减数列, 由于 $J \geq 0$, 有下界, 因此一定收敛.

并且我们可知, J 的值由簇的分类 i 和簇中心 $\boldsymbol{\mu}_i$ 唯一确定, 而 $\boldsymbol{\mu}_i = \frac{\sum_{i=1}^{K} \gamma_{ij} \boldsymbol{x}_i}{\sum_{i=1}^{K} \gamma_{ij}}$, 因此 $\boldsymbol{\mu}_i$ 也由 i 唯一确定, 也即 J 由 i 唯一确定, 其中 $j = 1, \cdots, k$.

由于样本 $\boldsymbol{x}_j$ 是有限个的, 因此 i 的划分方式是有限个的, 也就是 J 的取值是离散的有限个的值, 再由 J 是严格单调递减数列且有下界可知, 一定会有一个最小值 $J_{\min} \leq J$, 对于任何一个 J 值来说. 因此 J 一定会在 $J_{\min}$ 的时候停止, 与假设矛盾.

因此算法会在有限步内停止, 即能够收敛.

2 习题二

(a)

由线性回归的模型假设

$$y = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\beta} + \epsilon$$

可得平方误差

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})^2$$

则线性回归任务可以表示为优化问题

$$\underset{\boldsymbol{\beta}}{argmin} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})^2$$

(b)

使用 $\boldsymbol{X}$ 和 $\boldsymbol{y}$ 重写优化问题即有

$$\underset{\boldsymbol{\beta}}{argmin} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$$

(c)

令平方误差 $E = \sum_{i=1}^{n} \epsilon_i^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$

为了最小化 E, 我们对 β 求导且令其等于零向量有

$$\frac{\partial E}{\partial \boldsymbol{\beta}} = 2\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}) = \boldsymbol{0}$$

由于 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 是可逆的, 则有最优

$$\boldsymbol{\beta}^* = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

(d)

由于 $d > n$, 且我们知道 $\boldsymbol{X}$ 是一个 $n \times d$ 矩阵, 则其秩 $\mathrm{rank}(\boldsymbol{X}) \leq n < d$.

由矩阵的性质可知 $\mathrm{rank}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}) = \mathrm{rank}(\boldsymbol{X}) \leq n < d$.

而我们又知道 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 是一个 $d \times d$ 的矩阵, 因此 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 必然不满秩, 因此 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 不可逆.

(e)

该正则项带来的影响为减少模型的复杂度. 在实际问题中, 我们常常会遇到示例相对较少, 而特征较多的情况, 这种情况下 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 不一定可逆, 因此无法获得唯一的模型参数, 会有多个模型能够" 完美" 拟合训练集中的所有样例. 在加入正则化项之后, 由于正则化表示了对模型的一种偏好, 可以对模型的复杂度进行约束, 因此相对于在多个训练集上表现痛的预测结果的模型中选出模型复杂度最低的一个.

(f)

加入正则化项后得到的岭回归的优化问题为

$$\underset{\boldsymbol{\beta}}{argmin}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})+\lambda\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta}$$

为了最小化 E, 我们对 $\boldsymbol{\beta}$ 求导并令其等于零向量可得

$$\frac{\partial E}{\partial \boldsymbol{\beta}}=2\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{y})+2\lambda\boldsymbol{\beta}=\mathbf{0}$$

解得最优

$$\boldsymbol{\beta}^{*}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

(g)

当 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 不可逆时, 我们无法求解普通线性回归, 也即无法获得唯一的模型参数, 会有多个模型能够" 完美" 拟合训练集中的所有样例. 加入岭回归正则项后, $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I}$ 几乎总是可逆的, 进而可以求解.

在加入正则化项之后, 由于正则化表示了对模型的一种偏好, 可以对模型的复杂度进行约束, 因此相对于在多个训练集上表现痛的预测结果的模型中选出模型复杂度最低的一个.

(h)

如果 $\lambda=0$, 岭回归退化为普通线性回归.

如果 $\lambda=\infty$, 则优化问题变为 $\underset{\boldsymbol{\beta}}{argmin}\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta}$, 进而只能解出 $\boldsymbol{\beta}=\mathbf{0}$.

(i)

不能, 因为联合优化的话岭回归会退化为普通线性回归, 即一定有 $\lambda = 0$.

我们可以使用反证法来说明:

假设我们联合优化后得到最优 $\boldsymbol{\beta}^*$ 和非零的最优 λ^*, 对应的岭回归损失函数值为

$$E^* = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*)^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*) + \lambda^* \boldsymbol{\beta}^{*\mathrm{T}} \boldsymbol{\beta}^*$$

但是我们令 $\lambda = 0$ 有

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*)^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*) < (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*)^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*) + \lambda^* \boldsymbol{\beta}^{*\mathrm{T}} \boldsymbol{\beta}^* = E^*$$

则与 E^* 是最小岭回归损失矛盾, 因此一定有 $\lambda = 0$.

3 习题三

(a)

下标	类别标记	得分	查准率 P	查全率 R	AUC-PR	AP
0	-	-	1.0000	0.0000	-	-
1	1	1.0	1.0000	0.2000	0.2000	0.2000
2	2	0.9	0.5000	0.2000	0.0000	0.0000
3	1	0.8	0.6667	0.4000	0.1167	0.1333
4	1	0.7	0.7500	0.6000	0.1417	0.1500
5	2	0.6	0.6000	0.6000	0.0000	0.0000
6	1	0.5	0.6667	0.8000	0.1267	0.1333
7	2	0.4	0.5714	0.8000	0.0000	0.0000
8	2	0.3	0.5000	0.8000	0.0000	0.0000
9	1	0.2	0.5556	1.0000	0.1056	0.1111
10	2	0.1	0.5000	1.0000	0.0000	0.0000
-	-	-	-	-	0.6906	0.7278

(b)

如表格所示. 由于 $AUC-PR$ 和 AP 都是对 PR 曲线的总结, 因此它们的值确实彼此相似. 但是由于所采取的计算方法不同, 精度估计不同, 最后得到的结果也会有所差异, 例如此处的 AP 就比 $AUC-PR$ 要大 0.0372.

同理我们也可以通过数学推导的方式证明相似:

$$AUC\_PR = \sum_{i=1}^{n}(r_i - r_{i-1})\frac{p_i + p_{i-1}}{2}$$

$$AP = \sum_{i=1}^{n}(r_i - r_{i-1})p_i$$

两式相减可得

$$\begin{aligned} AP - AUC\_PR &= \sum_{i=1}^{n}(r_i - r_{i-1})p_i - \sum_{i=1}^{n}(r_i - r_{i-1})\frac{p_i + p_{i-1}}{2} \\ &= \sum_{i=1}^{n}\frac{1}{2}(r_i - r_{i-1})(p_i - p_{i-1}) \end{aligned}$$

可以看出 AP 总是比 $AUC-PR$ 大一点, 但是不会过分地大.

(c)

交换了第 9 行和第 10 行的类别标记之后, 新的 $AUC-PR$ 为 0.6794, 新的 AP 为 0.7167.

(d)

代码如下:

```
values = [1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]
labels = [1, 2, 1, 1, 2, 1, 2, 2, 1, 2]
# swap 9 and 10
# labels = [1, 2, 1, 1, 2, 1, 2, 2, 2, 1]

P = [1.0]
R = [0.0]
TPR = [0.0]
```

```
FPR = [0.0]

for i in range(1, len(values) + 1):
    P_counter = Counter(labels[:i])
    N_counter = Counter(labels[i:])
    TP = P_counter.get(1, 0)
    FP = P_counter.get(2, 0)
    FN = N_counter.get(1, 0)
    TN = N_counter.get(2, 0)
    P.append(TP / (TP + FP))
    R.append(TP / (TP + FN))

AUC_PR = [0.5 * (R[i] - R[i - 1]) * (P[i] + P[i - 1])
    for i in range(1, len(R))]
AUC_PR_SUM = sum(AUC_PR)
AP = [(R[i] - R[i - 1]) * P[i] for i in range(1, len(R))]
AP_SUM = sum(AP)

print('P:', ['%.4f' % f for f in P])
print('R:', ['%.4f' % f for f in R])
print('AUC_PR:', ['%.4f' % f for f in AUC_PR])
print('AUC_PR_SUM:', '%.4f' % AUC_PR_SUM)
print('AP:', ['%.4f' % f for f in AP])
print('AP_SUM:', '%.4f' % AP_SUM)
```

4 习题四

本实验探究 PCA 过程中是否中心化对主成分方向的影响。当未减去均值时，数据的主成分方向会受到均值向量的影响，尤其是当均值缩放因子较大时，第一主成分方向可能与均值向量高度相关。通过改变缩放因子 scale，观察未中心化 PCA 的第一主成分与均值向量、中心化 PCA 第一主成分之间的关系。

Python 代码实现

```
import numpy as np

def analyze_scale(scale_value):
    np.random.seed(0)
    avg = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
    scale = scale_value
    #生成数据
    data = np.random.randn(5000, 10) + np.tile(avg * scale, (5000, 1))
    #计算均值
    m = np.mean(data, axis=0)
    #未中心化SVD
    _, S, V = np.linalg.svd(data)
    e1 = V[0, :]
    # 中心化SVD
    newdata = data - np.tile(m, (5000, 1))
    _, S, V = np.linalg.svd(newdata)
    new_e1 = V[0, :]
    #标准化向量
    avg_norm = avg - np.mean(avg)
    avg_norm = avg_norm / np.linalg.norm(avg_norm)

    e1 = e1 - np.mean(e1)
    e1 = e1 / np.linalg.norm(e1)
    print("right eigenvectors :",new_e1)
    new_e1 = new_e1 - np.mean(new_e1)
    new_e1 = new_e1 / np.linalg.norm(new_e1)
    #计算相关性
    corr1 = np.abs(np.dot(avg_norm, e1))
    corr2 = np.abs(np.dot(e1, new_e1))

    return corr1, corr2, new_e1
```

```
#测试不同的scale值
scale_values = [1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
results = [analyze_scale(s) for s in scale_values]
corr1_values = [r[0] for r in results]
corr2_values = [r[1] for r in results]
print("avg and vec w/o centralization:",corr1_values)
print("vec corr between w/ and w/o centralization:",corr2_values)
```

结果分析

运行代码输出如下：

```
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
right eigenvectors : [-0.31324317  0.28179063  0.45941853  0.45579333  0.00144348 -0.06
 -0.57780112 -0.19381403 -0.14049596  0.08838333]
avg and vec w/o centralization: [0.9999929768660101, 0.999973699337526, 0.9995142752248
vec corr between w/ and w/o centralization: [0.31830171349451614, 0.31870833824114775,
```

可观察到以下现象：

- 当 scale 较大时（如 scale=1），corr1 接近 1，说明未中心化的第一主成分与均值向量高度相关
- 随着 scale 减小，corr1 逐渐降低，当 scale 非常小时（如 0.0001），corr1 趋近于 0
- corr2 始终接近 1，说明中心化前后的主成分方向基本一致

解释

- 未中心化时，当 scale 较大，数据分布由均值向量主导，主成分方向与均值方向一致
- 中心化后，PCA 分析的是数据相对于均值的协方差结构，与 scale 缩放无关
- 当 scale 趋近于 0 时，未中心化数据近似零均值，两种方法结果趋于一致

5 习题五

(a)

当用 Givens 旋转矩阵的转置 $G(i,j,\theta)^T$ 左乘向量 $\mathbf{x} \in \mathbb{R}^m$ 时，结果向量 $\mathbf{y} = G(i,j,\theta)^T\mathbf{x}$ 仅在第 i 和 j 两个分量上发生变化，其余分量保持不变。具体而言：

$$y_i = c \cdot x_i - s \cdot x_j, \quad y_j = s \cdot x_i + c \cdot x_j,$$

其中 $c = \cos\theta$，$s = \sin\theta$，而其他分量满足 $y_k = x_k$（$k \neq i, j$）。

(b)

若希望 $y_j = 0$，需满足方程：

$$s \cdot x_i + c \cdot x_j = 0 \quad \text{且} \quad c^2 + s^2 = 1.$$

解得：

$$c = \frac{x_i}{r}, \quad s = -\frac{x_j}{r}, \quad \text{其中} \quad r = \sqrt{x_i^2 + x_j^2}.$$

当 x_i 和 x_j 均为零时，取 $c = 1$，$s = 0$。

(c)

无需显式计算 θ 或调用三角函数，可直接通过代数运算确定 c 和 s：

$$r = \sqrt{x_i^2 + x_j^2}, \quad c = \frac{x_i}{r}, \quad s = -\frac{x_j}{r}.$$

若 $r = 0$（即 $x_i = x_j = 0$），则设 $c = 1$，$s = 0$。

(d)

当 $G(i, j, \theta)^T$ 左乘矩阵 $A \in \mathbb{R}^{m \times n}$ 时，仅修改 A 的第 i 行和第 j 行。具体地，对每列 k，更新后的元素为：

$$A'_{ik} = c \cdot A_{ik} - s \cdot A_{jk}, \quad A'_{jk} = s \cdot A_{ik} + c \cdot A_{jk}.$$

通过选择适当的 i, j, c, s，可将 A_{jk} 置零。例如，若希望将元素 A_{jk} 置零，需选择 $i < j$ 并计算 c 和 s 使得 $A'_{jk} = 0$。每次 Givens 旋转的计算复杂度为 $O(n)$，因为需要更新两行的所有 n 个元素。

(e)

利用 Givens 旋转进行 QR 分解的步骤如下：

1. 初始化 Q 为单位矩阵，$R = A$。
2. 对 R 的每列 k，从下到上逐行消去非零元素。对于第 i 行 $(i > k)$，构造 Givens 旋转 $G(i, k, \theta)$ 使得 R_{ik} 置零。
3. 应用旋转矩阵到 R，并累积旋转到 Q：$R \leftarrow G^T R$，$Q \leftarrow Q \cdot G$。
4. 重复上述步骤直到 R 变为上三角矩阵。

最终，$A = QR$，其中 Q 为正交矩阵，R 为上三角矩阵。此方法通过逐次消元构造 QR 分解，适用于稀疏矩阵或需要数值稳定的场景。

6 习题六

(a)

由矩阵 2-范数的定义可知 $\|\boldsymbol{X}\|_2 = \sigma_1$, 且由矩阵的逆的性质可知 $\left\|\boldsymbol{X}^{-1}\right\|_2 = \frac{1}{\sigma_n}$, 因此有

$$\kappa_2(\boldsymbol{X}) = \|\boldsymbol{X}\|_2 \left\|\boldsymbol{X}^{-1}\right\|_2 = \frac{\sigma_1}{\sigma_n}$$

(b)

若我们想要求解 $\boldsymbol{Ax} = \boldsymbol{b}$, 则有 $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$.

我们要说明的就是, 在 $\kappa_2(\boldsymbol{A})$ 很大的情况下, 稍微改变 $\boldsymbol{A}$ 或 $\boldsymbol{b}$ 就会导致 $\boldsymbol{x}$ 有很大的改变.

推导可知 (本小问中的 $\|\cdot\|$ 均指 2-范数 $\|\cdot\|_2$)

$$\boldsymbol{Ax} = \boldsymbol{bA}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b} + \Delta\boldsymbol{b}\boldsymbol{A}\Delta\boldsymbol{x} = \Delta\boldsymbol{b}\Delta\boldsymbol{x} = A^{-1}\Delta\boldsymbol{b}$$

由上面的式子可得

$$\|\boldsymbol{b}\| \le \|\boldsymbol{A}\| \|\boldsymbol{x}\| \ \|\Delta\boldsymbol{x}\| \le \left\|\boldsymbol{A}^{-1}\right\| \|\Delta\boldsymbol{b}\|$$

两式相乘再除以 $\|\boldsymbol{b}\| \|\boldsymbol{x}\|$ 可得

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \|\boldsymbol{A}\| \left\|\boldsymbol{A}^{-1}\right\| \frac{\|\Delta\boldsymbol{b}\|}{\boldsymbol{b}} = \kappa_2(\boldsymbol{A})\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$$

同理我们对 $\boldsymbol{A}$ 进行扰动变为 $\boldsymbol{A} + \Delta\boldsymbol{A}$ 可得

$$(\boldsymbol{A} + \Delta\boldsymbol{A})(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{bA}\Delta\boldsymbol{x} = -\Delta\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x})\Delta\boldsymbol{x} = -\boldsymbol{A}^{-1}\Delta\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x})$$

因此我们使用范数不等式并两边除以 $\|\boldsymbol{x} + \Delta\boldsymbol{x}\|$ 有

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x} + \Delta\boldsymbol{x}\|} \le \|\boldsymbol{A}\| \left\|\boldsymbol{A}^{-1}\right\| \frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|} = \kappa_2(\boldsymbol{A})\frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}$$

可以看出, 当有较小的扰动 $\Delta\boldsymbol{A}$ 或者 $\Delta\boldsymbol{b}$ 的时候, 尤其是能够取得等号的时候, 均会带来较大的 $\Delta\boldsymbol{x}$, 即较小的输入变换就会导致较大的输出变化. 这种变化对我们使用计算机进行一定精度的矩阵计算尤为不利.

例如用 Matlab 求解下列方程时:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0.999 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 1 \\ 0.999 \end{bmatrix} \Rightarrow \boldsymbol{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

条件数 $\kappa_2(\boldsymbol{A}) = 3998$, 可以看出是一个较大的数.

扰动 $\boldsymbol{A}$ 后求解 $\boldsymbol{x}$:

$$\begin{bmatrix} 1.001 & 1.001 \\ 1.001 & 0.999 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 1 \\ 0.999 \end{bmatrix} \Rightarrow \boldsymbol{x} = \begin{bmatrix} 0.499 \\ 0.500 \end{bmatrix}$$

扰动 $\boldsymbol{b}$ 后求解 $\boldsymbol{x}$:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0.999 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 1.001 \\ 0.999 \end{bmatrix} \Rightarrow \boldsymbol{x} = \begin{bmatrix} -0.999 \\ +2.000 \end{bmatrix}$$

可以看出, 无论是对 $\boldsymbol{A}$ 还是 $\boldsymbol{b}$ 做一个轻微的扰动, 均会导致解出来的 $\boldsymbol{x}$ 产生较大的变化, 因此这个线性系统是病态的.

(c)

对于正交矩阵 $\boldsymbol{W}$, 我们有 $\boldsymbol{W}^{-1} = \boldsymbol{W}^{\mathrm{T}}$, 且 $\boldsymbol{W}^{\mathrm{T}}$ 与 $\boldsymbol{W}$ 有相同的特征值, 以及 $\boldsymbol{W}$ 的奇异值 (特征值) 绝对值为 1. 因此有

$$\kappa_2(\boldsymbol{X}) = \|\boldsymbol{W}\|_2 \left\|\boldsymbol{W}^{-1}\right\|_2 = \|\boldsymbol{W}\|_2 \left\|\boldsymbol{W}^{\mathrm{T}}\right\|_2 = (\|\boldsymbol{W}\|_2)^2 = 1$$

因此正交矩阵是良态的, 有较小的条件数.

7 习题七

(2)

本题代码如下, 运行后, 终端会直接显示具体的数值结果:

```
 import numpy as np
from sklearn.decomposition import PCA

# 加载特征
features = np.load('cls_features.npy')
```

```
# 执行PCA
pca = PCA(n_components=0.9)
pca.fit(features)

# 获取结果
n_components = pca.n_components_
original_dim = features.shape[1]
percentage = (n_components / original_dim) * 100

print(f'保留90%方差所需维度: {n_components}')
print(f'占比原始维度({original_dim})的百分比: {percentage:.2f}%')
```

ViT-Tiny 的原始维度: $d = 192$（通过 $features.shape[1]$ 验证）
保留的维度数：运行后输出为 k, $k = 70$
百分比： $(k/192) * 100\% = 70/192 = 36.46\%$