一. (30 points) **机器学习导论复习题 (前八章)**

高级机器学习的课程学习建立在机器学习导论课程的基础之上，从事机器学习行业相关科研工作需要较为扎实的机器学习背景知识。下面的题目则对机器学习基础知识进行复习。

1. (10 points) 在现实中的分类任务通常会遇到类别不平衡问题，即分类任务中不同类别的训练样例数目类别差别很大，很可能导致模型无法学习。(a) 请介绍类别不平衡学习常用的策略。(b) 假定当前数据集中每个样本点包含了 $(\boldsymbol{x}_i, y_i, p_i)$，其中 $\boldsymbol{x}_i, y_i, p_i$ 分别表示第 i 个样本的特征向量，类别标签，和样本的重要程度，$0 \leq p_i \leq 1$。对于 SVM，任意误分样本点 $\boldsymbol{x}_i$ 的惩罚用 p_i 代替，请在西瓜书 p130 页公式 (6.35) 的基础上修改出新的优化问题，并给出对偶问题的推导。
2. (20 points) 通常情况下，模型会假设训练样本所有属性变量的值都已被观测到，但现实中往往会存在属性变量不可观测，例如西瓜根蒂脱落了，就无法观测到该属性值，此时问题就变成了有“未观测”变量的情况下，对模型参数进行估计。EM(Expectation-Maximization) 算法为常用的估计参数隐变量的方法。(a) 假设有 3 枚硬币，分别记作 A，B，C。这些硬币正面出现的概率分别是 a, b, c. 进行如下投掷实验：先投掷硬币 A，根据其结果选出硬币 B 或者硬币 C，正面选硬币 B，反面选硬币 C；然后投掷选出的硬币，投掷硬币的结果，出现正面记作 1，出现反面记作 0；独立地重复 n 次实验。假设只能观测到投掷硬币的结果，不能观测投掷硬币的过程。问如何估计三硬币正面出现的概率。请基于 EM 算法思想详细地写出 E 步和 M 步的推导步骤。(b) 经典的聚类算法 K-means 就是 EM 算法的一种特殊形式，K-means 也被称为 hard EM。请使用 EM 算法的思想解释 K-means，并对比 K-means 和 EM 算法的不同之处。

解：

1. (a): 常用策略有“欠采样”，“过采样”，“阈值移动”等。
“欠采样”：指的是，在正反例数量不平衡的情况下，采取减少一些反例的方法来进行平衡，之后再进行学习
“过采样”：与欠采样不同的是采用增加正例的方式来进行平衡，之后再进行学习
“阈值移动”：假设正例数目为 n^+, 反例数目为 n^-，分类函数为 y
先让模型基于原数据进行学习，在预测类别的时候基于 $\frac{y}{1-y} \times \frac{n^-}{n^+} \geq 1$ 对样本进行分类
(b): 那么，根据西瓜书 p130 页公式 (6.35) 有优化问题:

$$\begin{aligned} \min_{w,b} \frac{1}{2}||w||^2 + \sum_{i=1}^{m} p_i \xi_i \\ s.t. \xi_i \geq 0, i = 1, 2, ..., m \\ y_i(w^T x_i + b) \geq 1 - \xi_i \end{aligned}$$

对偶问题转化:

$$\begin{aligned} L(w, b, \alpha, \mu, \xi) &= \frac{1}{2}||w||^2 + \sum_{i=1}^{m} p_i \xi_i + \sum_{i=1}^{m} \alpha_i (1 - \xi_i - y_i(w^T x_i + b)) - \sum_{i=1}^{m} \mu_i \xi_i \\ \frac{\partial L}{\partial w} &= w - \sum_{i=1}^{m} \alpha_i y_i x_i \\ \frac{\partial L}{\partial b} &= \sum_{i=1}^{m} \alpha_i y_i \\ \frac{\partial L}{\partial \xi_i} &= p_i - \alpha_i - \mu_i \end{aligned}$$

令各项偏导为零可知:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0$$
$$p_i = \alpha_i + \mu_i$$

所以代入后问题转换为

$$\begin{aligned} L(w,b,\alpha,\mu,\xi) = &\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{m}(\alpha_i+\mu_i)\xi_i + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i\xi_i - \\ &\sum_{i=1}^{m}\alpha_i y_i (\sum_{i=1}^{m}\alpha_i y_i x_i)^T x_i + b\sum_{i=1}^{m}\alpha_i y_i - \sum_{i=1}^{m}\mu_i\xi_i \\ = &\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j \end{aligned}$$

所以对偶问题是:

$$\max_{\alpha} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$$
$$s.t. \sum_{i=1}^{m}\alpha_i y_i = 0$$
$$0 \le \alpha_i \le \xi_i$$

2. (a): 把最终的观测结果设为 $y = (y_1, y_2, ..., y_n)$, 中途硬币 A 的投掷结果设为隐变量 z, 构建模型的参数 $\theta = (a, b, c)$
对于第 i 次投掷硬币的结果，可以推出概率表达式:

$$\begin{aligned} p(y_i|\theta) &= \sum_z p(z|\theta)p(y_i|\theta,z) \\ &= ab^{y_i}(1-b)^{1-y_i} + (1-a)c^{y_i}(1-c)^{1-y_i} \end{aligned}$$

$$p(y_i, z=0|\theta) = (1-a)c^{y_i}(1-c)^{1-y_i}$$
$$p(y_i, z=1|\theta) = ab^{y_i}(1-b)^{1-y_i}$$
$$p(z=0|y_i,\theta) = \frac{p(y_i,z=0|\theta)}{p(y_i|\theta)} = \frac{(1-a)c^{y_i}(1-c)^{1-y_i}}{ab^{y_i}(1-b)^{1-y_i} + (1-a)c^{y_i}(1-c)^{1-y_i}}$$
$$p(z=1|y_i,\theta) = \frac{p(y_i,z=1|\theta)}{p(y_i|\theta)} = \frac{ab^{y_i}(1-b)^{1-y_i}}{ab^{y_i}(1-b)^{1-y_i} + (1-a)c^{y_i}(1-c)^{1-y_i}}$$

为了方便，令

$$p(z=1|y_i,\theta) = 1-\gamma_i$$
$$p(z=0|y_i,\theta) = \gamma_i$$

则 Q 函数可求得:

$$\begin{aligned} Q(\theta^{(0)}) &= \sum_{i=1}^{n}[p(z=0|y_i,\theta)\ln p(y_i,z=0|\theta) + p(z=1|y_i,\theta)\ln p(y_i,z=1|\theta)] \\ &= \sum_{i=1}^{n}\gamma_i \ln ab^{y_i}(1-b)^{1-y_i} + (1-\gamma_i)\ln(1-a)c^{y_i}(1-c)^{1-y_i} \end{aligned}$$

到这里，EM 算法的 E 步求取隐变量的概率分布也就完成了
接下来进行 M 步计算:

$$\theta^1 = \arg\max_{\theta} Q(\theta^{(0)})$$

$$\begin{aligned} \frac{\partial LL(\theta)}{\partial a} &= \sum_{i=1}^{n}\gamma_i\frac{b^{y_i}(1-b)^{1-y_i}}{ab^{y_i}(1-b)^{1-y_i}} + (1-\gamma_i)\frac{-c^{y_i}(1-c)^{1-y_i}}{(1-a)c^{y_i}(1-c)^{1-y_i}} \\ &= \sum_{i=1}^{n}\frac{\gamma_i}{a} - \frac{1-\gamma_i}{1-a} \\ &= \sum_{i=1}^{n}\frac{(1-a)(\gamma_i) - a(1-\gamma_i)}{a(1-a)} \\ &= \sum_{i=1}^{n}\frac{\gamma_i - a}{a(1-a)} \\ \frac{\partial LL(\theta)}{\partial b} &= \sum_{i=1}^{n}\gamma_i\frac{ay_ib^{y_i-1}(1-b)^{1-y_i} - ab^{y_i}(1-y_i)(1-b)^{-y_i}}{ab^{y_i}(1-b)^{1-y_i}} \\ &= \sum_{i=1}^{n}\gamma_i\frac{ab^{y_i}(1-b)^{1-y_i}[\frac{y_i}{b} - \frac{1-y_i}{1-b}]}{ab^{y_i}(1-b)^{1-y_i}} \\ &= \sum_{i=1}^{n}\gamma_i\frac{y_i - by_i - b + by_i}{b(1-b)} \\ &= \sum_{i=1}^{n}\gamma_i\frac{y_i - b}{b(1-b)} \\ \frac{\partial LL(\theta)}{\partial c} &= \sum_{i=1}^{n}(1-\gamma_i)\frac{(1-a)[y_ic^{y_i-1}(1-c)^{1-y_i} - c^{y_i}(1-y_i)(1-c)^{-y_i}]}{(1-a)c^{y_i}(1-c)^{1-y_i}} \\ &= \sum_{i=1}^{n}(1-\gamma_i)\frac{(1-a)c^{y_i}(1-c)^{1-y_i}[\frac{y_i}{c} - \frac{1-y_i}{1-c}]}{(1-a)c^{y_i}(1-c)^{1-y_i}} \\ &= \sum_{i=1}^{n}(1-\gamma_i)\frac{y_i - c}{c(1-c)} \end{aligned}$$

令各项偏导数为 0 可以求得:

$$\frac{\partial LL(\theta)}{\partial a} = 0$$
$$\sum_i^n \frac{\gamma_i^{(0)}}{a(1-a)} = \sum_i^n \frac{a}{a(1-a)}$$
$$a^{(1)} = \frac{1}{n}\sum_i^n \gamma_i^{(0)}$$

$$\frac{\partial LL(\theta)}{\partial b} = 0$$
$$\sum_i^n \frac{\gamma_i^{(0)} y_i}{b(1-b)} = \sum_i^n \frac{\gamma_i^{(0)} b}{b(1-b)}$$
$$b^{(1)} = \frac{\sum_i^n \gamma_i^{(0)} y_i}{\sum_i^n \gamma_i^{(0)}}$$

$$\frac{\partial LL(\theta)}{\partial c} = 0$$
$$\sum_i^n \frac{(1-\gamma_i^{(0)}) y_i}{c(1-c)} = \sum_i^n \frac{(1-\gamma_i^{(0)}) c}{c(1-c)}$$
$$c^{(1)} = \frac{\sum_i^n (1-\gamma_i^{(0)}) y_i}{\sum_i^n (1-\gamma_i^{(0)})}$$

至此，第一轮参数更新就完成了。
按照上述过程继续迭代更新参数直至收敛即可完成 EM 算法的流程。
(b): 对于 K-means 算法而言，其中 EM 算法的 E 步与 K-means 算法中对于每一个点 x_i 找到其最近的聚类中心点 μ_{y_i} 是等价的
EM 算法的 M 步与 K-means 算法中的求新的聚类中心点是等价的
K-means 与 EM 算法的不同在于 K-means 算法的过程是在给每一个 observation 分配一个聚类类别，而 EM 算法的运作过程是在寻找一个聚类类别的似然函数
并且 K-means 对于类别的分类是 hard 的，是一种 0-1 选择，但是 EM 算法得出的是一个概率分布，举个例子而言就是可能有 0.3 的概率分配给 C_1 类，可能有 0.7 的概率分配给 C_2 类

二. (25 points) **主成分分析**

主成分分析 (Principal Component Analysis，PCA) 是一种经典的无监督降维技术，可以有效减少数据维度，避免维度灾难。实际上，涉及 PCA 的算法有非常多，下面的题目将逐步引入更多关于 PCA 的内容。

1. (5+5 points) 关于 PCA，教材中给出了最近重构性和最大可分性两种推导方法，但是该方法将多个主成分在一起推导。实际上，有另外一种 Step-by-step 的推导方法更为具体。假设数据矩阵 $X \in \mathcal{R}^{n\times d}$ 包含 n 个 d 维度的样本，每个样本记作 $x_i \in \mathcal{R}^d$。下面基于 Step-by-step 的最大可分性进行推导。最大可分性的假设偏好是：样本在低维空间尽可能分散。(a) 假设选取第一个主成分为 $w \in \mathcal{R}^d$，需要满足 $\|w\|_2^2 = 1$，那么样本投影到该主成分的投影点为 $w^T x_i$，然后我们需要最大化投影点之间的方差，试写出具体的优化目标，并分析其与瑞利商 (Rayleigh quoient) 的关系。可假设数据已经中心化。(b) 在选取第一个主成分 w 之后，需要求解第二个主成分 v，要满足和第一个主

成分向量正交，即 $v^T w = 0$，此时可以考虑将样本 x_i 分解为两个成分：沿着 w 的向量和垂直于 w 的向量。最后只需要对于垂直的部分选取第二个主成分即可。试给出具体的分解方法以及后续选取第二个主成分的推导过程。

2. (5+5 points) 假设 PCA 得到的映射矩阵 (主成分组成的矩阵) 为 $W \in \mathcal{R}^{d\times d'}$，那么对数据矩阵 $X \in \mathcal{R}^{n\times d}$ 降维的过程是：$XW \in \mathcal{R}^{n\times d'}$。该过程可以看作是神经网络中不带有偏置 (bias) 的一层全连接映射。那么：(a) 基于最近重构性的 PCA 推导方法和 AutoEncoder 有什么关系？试分析二者的区别和联系 (可以从公式、优化、实验效果等角度进行分析)。(b) 一般地，在深度神经网络中，对于全连接层会加入正则化项，例如二范数正则化 $\|W\|_2^2$，在 PCA 中是否可以同样地对 W 施加正则化项呢？试给出具体的优化目标以及大概如何求解。(可参考 Sparse PCA 相关内容，只需说出求解优化问题的方法，无需给出具体求解算法和过程)。
3. (5 points) (任选一题) 上题谈到了 PCA 和深度神经网络，我们知道深度神经网络一般基于梯度自动回传来进行反向传播，其自动梯度计算过程在 PyTorch、Tensorflow 等工具包中已经被实现。试问：(a) 请调研 sklearn 中实现的 SVD 的方法，试比较其提供的 FullSVD、TruncatedSVD、RandomizedSVD 等 SVD 的区别，如果有实验效果对比图 (性能、运行效率) 则更佳。(b) 试问在 PyTorch 中是否可以对 SVD 进行自动计算梯度，如有，请简单介绍其原理。

解：

1. (a): 最大化投影点之间的方差可以得知

$$\max Var(\boldsymbol{w}^T x_i) = \max \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T x_i - \overline{\boldsymbol{w}^T x_i})^2$$

因为数据已经中心化，所以 $\overline{\boldsymbol{w}^T x_i} = 0$
所以问题转化为

$$\max_{||\boldsymbol{w}||_2^2=1} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T x_i)^2$$

$$\max_{||\boldsymbol{w}||_2^2=1} \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w}$$

因为 $||\boldsymbol{w}||_2^2 = 1$，所以 $\boldsymbol{w}^T\boldsymbol{w} = 1$
所以上述问题可以等价转换为

$$\max_{||\boldsymbol{w}||_2^2=1} \frac{\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}}$$

而优化问题的优化目标就是在优化瑞利商的一个等价形式。
(b): 对于一个样本点 x_i, 可以分解为

$$x_i = \boldsymbol{w}^T x_i \times \boldsymbol{w} + (x_i - \boldsymbol{w}^T x_i \times \boldsymbol{w})$$

容易推导得

$$\boldsymbol{w}^T(x_i - \boldsymbol{w}^T x_i \times \boldsymbol{w}) = \boldsymbol{w}^T x_i - \boldsymbol{w}^T x_i \boldsymbol{w}^T\boldsymbol{w} = 0$$

因此，样本 x_i 已经分解为了互相垂直的两个部分
那垂直部分为

$$\boldsymbol{X}' = \boldsymbol{X} - \boldsymbol{X}\boldsymbol{w}\boldsymbol{w}^T$$

那么第二主成分为

$$v = \arg\max \frac{v^T \boldsymbol{X}^T \boldsymbol{X} v}{v^T v}$$

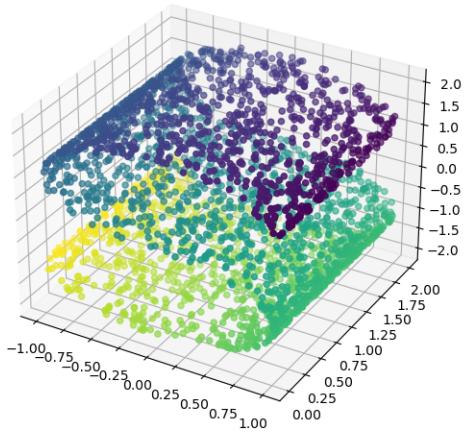
2. (a): 基于最近重构性的 PCA 的优化目的在于使得降维后的样本 x' 与降维前的样本 x 之间的距离最小化。
从公式上看，最近重构性的 PCA 在优化:

$$\min \sum_{i=1}^{m} ||x' - x||_2^2$$

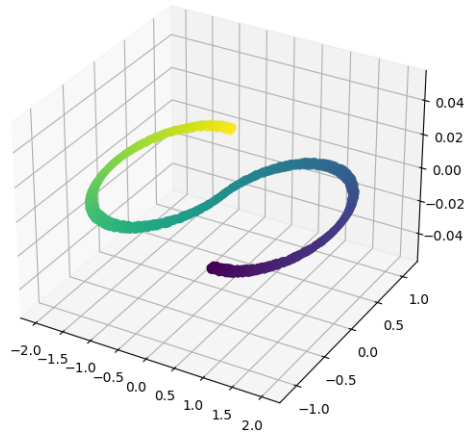
而 Autoencoder 的过程是将一个样本 $\boldsymbol{X} = (x_1, x_2, ..., x_m)$ 通过 encoder 转化为 $\boldsymbol{Y}$, 再通过 decoder 转化为 $\mathcal{X}$, 其优化目标是让经过 encoder-decoder 作用之后的 $\mathcal{X}$ 与原始 $\boldsymbol{X}$ 之间的距离最小化。
从公式上看，Autoencoder 在优化:

$$\min ||\mathcal{X} - \boldsymbol{X}||^2$$

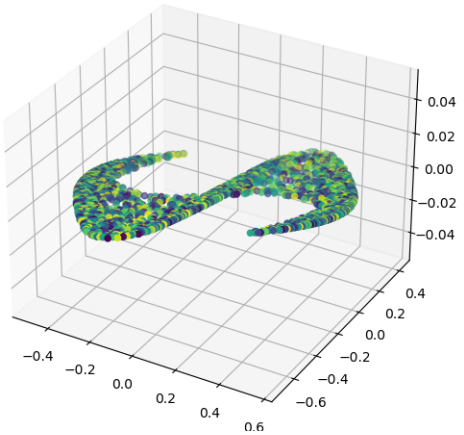
这两个优化问题在本质上其实是等价的。
单单对于降维这个目标来说，可以选择通过 encoder 就完成降维或者通过 encoder-decoder 来完成降维。
但是区别在于 Autoencoder 最终的降维可以处理非线性的部分，而 PCA 只能进行线性降维，这个特点也会在下面的实验结果中有所体现。
Autoencoder 通过一个单独的 layer 并利用线性的转换函数就可以基本实现 PCA 的效果, 在实验中通过简单的神经网络构建，选择线性函数以及 tanh 函数作为激励函数。
本实验的降维效果基于 S 型数据 (make_s_curve):
原图:



PCA 处理之后:



AutoEncoder 处理之后:



可见 PCA 降维处理之后就是一条曲线，而 AutoEncoder 处理之后就是一个带有曲度的面。
(b): 在 PCA 中也可以对 W 施加正则项。
根据 Sparse Principal Component Analysis 的相关内容可以给出一个具体的优化问题:

$$\hat{\beta} = (1+\lambda_2)\arg\min_{\beta} |\boldsymbol{Y} - \sum_{j=1}^{p} \boldsymbol{X_j}\beta_j| + \lambda_2 \sum_{j=1}^{p} |\beta_j|^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j|$$

求解这个问题需要用到 Sparse Principal Component Analysis 这篇 paper 中提到的算法:
1)：令 α 为样本数据 $\boldsymbol{V}$ 中的前 k 个主成分
2)：对于每一个给定的 α, 将上面已经提及的优化问题等价转化为:

$$\beta_j = \arg\min_{\beta^*} \beta^{*T}(\boldsymbol{X}^T\boldsymbol{X} + \lambda)\beta^* - 2\alpha_j^T \boldsymbol{X}^T\boldsymbol{X}\beta^* + \lambda_{1,j}|\beta^*|_1$$

3)：对于每一个给定的 β, 进行奇异值分解 $\boldsymbol{X}^T\boldsymbol{X}\beta = UDV^T$, 并更新 $\alpha = UV^T$
4)：重复 2-3，直到 β 收敛
5)：将结果归一化:$\hat{V_j} = \frac{\beta_j}{|\beta_j|}, j = 1, ..., k$
至此，上述优化问题就可以得到一个通解。

3. (b): 可以，pytorch 中通过对于参数 $requires\_grad$ 的设置就可以设定是否需要自动计算梯度。
原理如下:
对于每一个 tensor，都有一个属性 ġrad_fn 用以记录在反向传播过程中导数的计算方式，并且有属性.is_leaf 来判定是否作为叶子结点在传播过程中进行更新, 并且有一个 next_function 属性指向下一个运算。
当某个 tensor e 进行相关运算的时候，就会执行 backward()，并且 $requires\_grad = True$ 的时候，系统遍历每一个.is_leaf 为 True 的叶子，按照.grad_fn 中记录的方法进行更新, 并通过 next_function 属性将结果送入下一层运算。
因此，当运算结束时，梯度也进行了自动更新。

三. (15 points) **降维与度量学习**

降维与度量学习包含多种算法，例如 PCA、NCA、LLE、MDS 等等。接下来的几个题目会拓展大家对这些算法的认知范围。下面三个小题中选做任意两道即可。

1. (5 points) 近邻成分分析 (Neighbourhood Component Anslysis, NCA) 是基于 KNN 分类器的有监督降维算法。其优化目标主要是：$f = \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} \sum_{j \in C_i} p_{ij}$，其中 $C_i = \{j | y_j = y_i\}$ 表示与第

i 个样本类别一样的下标集合，$p_{ij} = \frac{\exp\left(-\|Ax_i - Ax_j\|_2^2\right)}{\sum_{k\neq i}\exp\left(-\|Ax_i - Ax_k\|_2^2\right)}, j \neq i, p_{ii} = 0$ 表示将第 i 个数据和其余所有样本的近邻概率分布 (NN 分类过程)，距离越近其对应的 p_{ij} 越大，f 的目标则是最大化留一验证近邻分类的准确性。$A \in \mathcal{R}^{d' \times d}$ 是待优化的映射矩阵。试推导其梯度 $\frac{\partial f}{\partial A}$。

2. (5 points) 在自然语言处理领域，潜在语义分析 (Latent Semantic Analysis, LSA) 可以从文档-词矩阵中学习到文档表示、词表示，本质上也是对矩阵进行分解，试查阅相关资料，描述其具体步骤。并简述其与 PCA 的区别。
3. (5 points) 根据局部线性嵌入 (Locally Linear Embedding, LLE) 的算法流程，尝试编写 LLE 代码，可以基于 sklearn 实现，并在简单数据集 ("S" 型构造数据或 Mnist 等) 上进行实验，展示实验结果。

解：

1. 为方便书写，令 $\gamma_{ij} = \exp(-||Ax_i - Ax_j||_2^2)$

$$\frac{\partial f}{\partial A} = \sum_{i=1}^{n}\sum_{j\in C_i}\frac{\partial p_{ij}}{\partial A} = \frac{1}{(\sum_{k\neq i}\gamma_{ik})^2}[\frac{\partial \gamma_{ij}}{\partial A}\sum_{k\neq i}\gamma_{ik} - \gamma_{ij}\sum_{k\neq i}\frac{\partial \gamma_{ik}}{\partial A}]$$

$$\frac{\partial \gamma_{ij}}{\partial A} = -2A\gamma_{ij}(x_i - x_j)(x_i - x_j)^T$$

代入之后有

$$\begin{aligned}
\frac{\partial f}{\partial A} &= \sum_{i=1}^{n}\sum_{j\in C_i}\frac{1}{(\sum_{k\neq i}\gamma_{ik})^2}[-2A\gamma_{ij}(x_i - x_j)(x_i - x_j)^T\sum_{k\neq i}\gamma_{ik} + \gamma_{ij}\sum_{k\neq i}2A\gamma_{ik}(x_i - x_k)(x_i - x_k)^T] \\
&= \sum_{i=1}^{n}\sum_{j\in C_i}[\frac{-2A\gamma_{ik}(x_i - x_j)(x_i - x_j)^T}{\sum_{k\neq i}\gamma_{ij}} + \frac{2A\gamma_{ij}\sum_{k\neq i}(x_i - x_j)(x_i - x_j)^T}{\sum_{h\neq i}\gamma_{ih}\sum_{k\neq i}\gamma_{ik}}] \\
&= \sum_{i=1}^{n}\sum_{j\in C_i}[-2Ap_{ij}(x_i - x_j)(x_i - x_j)^T\gamma_{ij} + 2p_{ij}A(\sum_{k\neq i}p_{ik}(x_i - x_k)(x_i - x_k)^T)] \\
&= -2A\sum_{i=1}^{n}\sum_{j\in C_i}p_{ij}[(x_i - x_j)(x_i - x_j)^T - \sum_{k\neq i}p_{ik}(x_i - x_k)(x_i - x_k)^T]
\end{aligned}$$

2. 具体步骤:
(1): 首先，LSA 需要对文档进行分析，构建一个 document-term matrix, 用来表述 terms 的出现频率。
(2): 其次，LSA 希望找到一个 document-term matrix 的低阶近似来减少计算开销、消除部分噪音、减少一些稀疏性。这一步会将部分表达意思相近的 dimensions 的信息进行组合表示。
(3): 带着这个目的，LSA 会对于这个 document-term matrix 进行建模，用数来表示频率信息，进行奇异值分解, 并进行降维操作，完成低阶近似。
(4): 对于低阶近似后的结果进行一系列的 query、compare，构建一个潜在语义空间，对于每一个 query，比较原文与结果所传达出的语义信息。
区别:
(1)LSA 是一种明确指定的分析、还原文本的方式，PCA 是一种通用的文本分析方式。
(2) 在 LSA 中，上下文主要通过 document-term matrix 的数字形式传达信息，PCA 主要是通过协方差矩阵传达信息。这也表示 LSA 在寻找一个最佳的线性子空间，而 PCA 在寻找一个并行的线性子空间。

3. 考虑选用 S 型构造数据作为 dataset，基于 sklearn 实现 lle。
调用 S 型构造数据需要执行:

```
from sklearn.datasets import make_s_curve
```

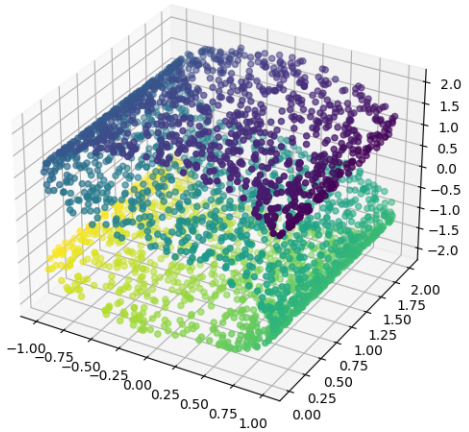
lle 方法需要调用 sklearn 自带的方法:

```
from sklearn.manifold import LocallyLinearEmbedding
```

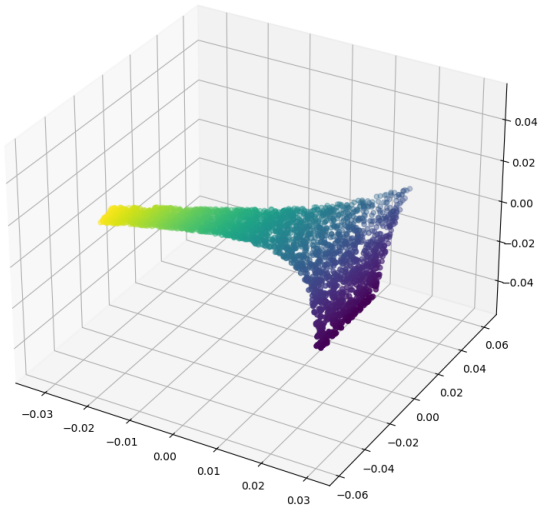
对于 make_s_curve 的参数设置如下，并不再进行改变:

```
make_s_curve(n_samples=3000 ,random_state=2021)
```

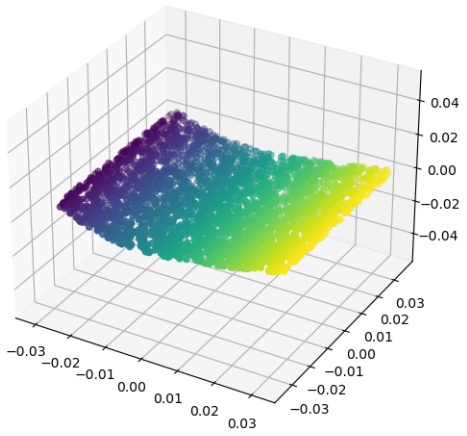
先行展示原始三维数据的图形:



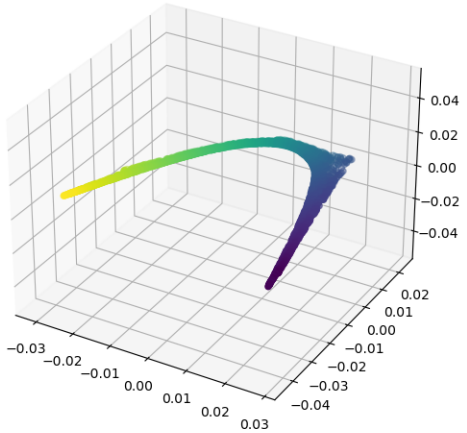
在参数为 $n\_components = 3, n\_neighbors = 10, random\_state = 2021$ 的 LLE 处理之后的图形为:



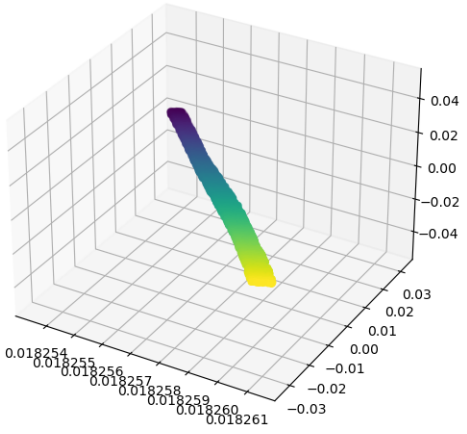
可以发现降维去除了横向的一维数据。
在换用了 the modified locally linear embedding algorithm 后，图像发生了明显的平面化:



在不改变算法的基础上，更改超参数正则项的值 $reg = 1e-2$ 后, 图像也发生了明显的聚集化:



更改超参数正则项的值 $reg = 1e-4$ 后, 图像不再呈现镰刀状



以上就是对于 sklearn 中 LLE 在 make_s_curve 数据集的简单结果展示。

四. (15 points) **特征选择基础**

Relief 算法中，已知二分类问题的相关统计量计算公式如下：

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,nh}^j\right)^2 + \operatorname{diff}\left(x_i^j, x_{i,\mathrm{nm}}^j\right)^2 \tag{1}$$

多分类的 Relief-F 算法的相关统计量计算公式如下：

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,\mathrm{nh}}^j\right)^2 + \sum_{l\neq k}\left(p_l \times \operatorname{diff}\left(x_i^j, x_{i,l,\mathrm{nm}}^j\right)^2\right) \tag{2}$$

其中 p_l 为第 l 类样本在数据集 D 中所占的比例。然而仔细观察可发现，二分类问题中计算公式的后一项 $\operatorname{diff}\left(x_i^j, x_{i,\mathrm{nm}}^j\right)^2$ 的系数为 1，多分类问题中后一项系数求和小于 1，即 $\sum_{l\neq k} p_l = 1 - p_k < 1$。基于这个发现，请给出一种 Relief-F 算法的修正方案。

解：

一种朴素的修正方法就是在计算比例时去除第 k 类的影响:

$$p_l' = \frac{p_l}{1-p_k}, l \neq k$$

将修正后的 p_l' 代入到原式当中，可以保证:

$$\sum_{l\neq k} p_l' = \frac{\sum_{l\neq k} p_l}{1-p_k} = 1$$

此时的 p_l' 仍然可以起到指示第 l 类样本的猜错近邻在 j 属性上的作用大小与其他所有可能猜错的类中的比例。

对修改后的 $p_l' \times \operatorname{diff}\left(x_i^j, x_{i,l,\mathrm{nm}}^j\right)^2$ 进行求和后，可以在保持系数为 1 的情况下，反映第 k 类样本以外的样本的估计平均。

五. (15 points) **特征选择拓展**

本题借助强化学习背景，主要探讨嵌入式选择在强化学习中的应用。强化学习可以看作一种最大化奖励 (也就是目标) 的机器学习方法，目的是学习到一个策略，使得执行这个策略获得的奖励值最大。基于 TRPO(一种强化学习方法) 的近似方法的近似问题如下

$$\begin{array}{ll} \max\limits_{\theta} & \left(\nabla L_{\theta_{\mathrm{old}}}(\theta)\right)^T (\theta - \theta_{\mathrm{old}}) \\ \text{s.t.} & \frac{1}{2}(\theta - \theta_{\mathrm{old}})^T H (\theta - \theta_{\mathrm{old}}) \leq \delta \end{array} \tag{3}$$

这里采用了参数化表示方法，其中 θ 表示新策略，θ_{old} 表示旧策略，方法需要通过策略的目标函数 $L_{\theta_{\mathrm{old}}}$ 来更新旧策略，最终目标是学习到最大化目标函数的新策略。这里要最大化的表达式可以对应理解为最小化损失函数，即类似于课本 252 页式 (11.5)。

如果将目标 L 分解为很多个子目标，即 $L = [L_1, L_2, \ldots, L_n]^T$，每个目标对应相应的权重 $w = [w_1, w_2, \ldots, w_n]^T$，新方法 (称为 ASR 方法) 的优化目标如下

$$\begin{array}{ll} \max_w & \max_\theta \left(\nabla \left(L^T w\right)\right)^T (\theta - \theta_{\mathrm{old}}) \\ \text{s.t.} & \frac{1}{2}(\theta - \theta_{\mathrm{old}})^T H (\theta - \theta_{\mathrm{old}}) \leq \delta \\ & \|w\|_1 = 1 \\ & w_i \geq 0, \quad i = 1, 2, \ldots, n \end{array} \tag{4}$$

问：

1. (10 points) 尝试分析 ASR 方法中加入 w 的 L1 范数约束的现实意义。(提示：不同目标对应的参数 w_i 是需要学习的参数。原目标 L 现由多个子目标组成，每个子目标的质量良莠不齐)
2. (5 points) 在 ASR 方法基础上提出的 BiPaRS 方法解除了 w 的 L1 范数这一限制，使得更多样 w 可以出现、更多种 L 可以被使用。结合这一点，论述特征选择需要注意的事项。

解：

1. (i): 如果没有 w 的 L1 范数约束，单单只有 $w_i \geq 0$ 的约束。
   那么对于不同的目标的学习参数 w_i 与 w_j 之间可能出现数值差距过大的情况，不妨假设 $w_i >> w_j$, 这将使得模型十分注重目标 L_i 的实现情况而极大忽略目标 L_j 的实现情况。
   在训练过程中，我们并不知道目标 L_i 与目标 L_j 的质量如何，如果目标 L_i 实际上对于整个模型的学习并没有什么好处但是权值却很大，需要经过大量的学习过程去修正，增加计算开销。并且这种侧重性可能只是某些训练数据的一个特点, 不具有泛化性, 这会不可避免地使得模型走向过拟合。
   通过引入 w 的 L1 范数约束，保证每一个 w_i 所起到的作用有限、可控，减少过拟合的可能性。
   (ii): 另外一方面，引入 L1 范数约束，使得不同目标的权重之和为 1, 也增强了模型的可解释性。对于不同的子目标，我们可以清楚地知道，学习的过程就是在优化、调整不同子目标对于整个目标的重要程度。
   (iii):L1 范数约束会增强稀疏性但又不至于过度稀疏，使得部分的正向子目标的重要性得以有效体现, 实现自动选择。

2. (i)ASR 方法中加入 w 的 L1 范数约束，但是在 BiPaRS 方法中又取消了 w 的 L1 范数约束, 而不同的特征选择可以训练出不同的模型应用于不同的问题。就像 BiPaRS 方法与 ASR 方法相比，对于不同的子目标就具有更强的包容性，优化目标的多样性也得以提升。
   由此可见，对于不同的任务，需要进行灵活的特征选择策略的调整。针对不同问题，进行不同的特征选取策略的选取来使得学习效果更佳。