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) 类别不平衡学习常用的策略有再缩放，常用的方法有：对训练集里的反类样例进行“欠采样”，再进行学习；对训练集里的正类样例进行“过采样”，再进行学习；直接基于原始训练集进行学习，但在预测时将再缩放的式子嵌入决策过程中，即“阈值移动”。

   (b) 用 p_i 表示惩罚，故优化目标可写为

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m} p_i \frac{1}{2}(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)) \tag{1}$$

   引入“松弛变量” $\xi_i = p_i\frac{1}{2}(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))$，由范围可知，$0 \leq \xi_i \leq 1$，可将 (1) 式改写为

$$\begin{aligned} &\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m} \xi_i \\ &s.t.\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \frac{2\xi_i}{p_i} \\ &\quad 0 \leq \xi_i \leq 1\ ,\ i = 1, 2, ..., m \end{aligned} \tag{2}$$

   (2) 式即新的优化问题，接下来推导其对偶问题。使用拉格朗日乘子法得到 (2) 式的拉格朗日函数，如下

$$\begin{aligned} L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) =& \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m} \xi_i \\ &+ \sum_{i=1}^{m} \alpha_i(1 - \frac{2\xi_i}{p_i} - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)) - \sum_{i=1}^{m} \mu_i\xi_i \end{aligned} \tag{3}$$

令 $L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu})$ 对 $\boldsymbol{w}, b, \boldsymbol{\xi}$ 的偏导为零可得

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \tag{4}$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i \tag{5}$$

$$C = \frac{2\alpha_i}{p_i} + \mu_i \tag{6}$$

将 (4)(5)(6) 式代入式 (3) 即可得到 (2) 式的对偶问题

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{7}$$

$$s.t. \ \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq \frac{p_i}{2} C \ , \ i = 1, 2, ..., m$$

2. (a) 要估计三硬币正面出现的概率，首先要使用 EM 算法来估计出 a、b、c。首先假设隐变量为 Z，其表示硬币 A 的结果，若硬币 A 正面朝上，则 $z_i = 1$，反之 $z_i = 0$；θ 为模型参数 a、b、c 的集合，初始化为 $\boldsymbol{\theta}^0$；Y 为观测变量，即硬币 B 或 C 投掷的结果。

对数似然可写为

$$\begin{aligned} LL(\boldsymbol{\theta} \mid Y, Z) &= lnP(Y, Z \mid \boldsymbol{\theta}) \\ &= ln(\prod_{i=1}^{n} p(y_i, z_i|\boldsymbol{\theta})) \\ &= \sum_{i=1}^{n} ln(p(y_i, z_i|\boldsymbol{\theta})) \end{aligned} \tag{8}$$

对 Z 计算期望，来最大化边际似然

$$\begin{aligned} LL(\boldsymbol{\theta} \mid Y) &= lnP(Y \mid \boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} lnP(y_i \mid \boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} ln \sum_{Z} P(y_i, z_i \mid \boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} ln(\sum_{Z} P(y_i|z_i, \boldsymbol{\theta}) P(z_i|\boldsymbol{\theta})) \end{aligned} \tag{9}$$

E 步推导如下

对数似然关于 Z 的期望为

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^t) &= E_{Z|X,\boldsymbol{\theta}^t} LL(\boldsymbol{\theta} \mid X, Z) \\ &= \sum_{i=1}^{n} \sum_{Z} p(z_i|y_i, \boldsymbol{\theta}^t) ln(p(y_i, z_i|\boldsymbol{\theta})) \\ &= \sum_{i=1}^{n} (p(z_i = 0|y_i, \boldsymbol{\theta}^t) ln(p(y_i, z_i = 0|\boldsymbol{\theta})) + p(z_i = 1|y_i, \boldsymbol{\theta}^t) ln(p(y_i, z_i = 1|\boldsymbol{\theta}))) \end{aligned} \tag{10}$$

其中

$$p(y_i, z_i = 0|\boldsymbol{\theta}) = (1-a)c^{y_i}(1-c)^{1-y_i} \tag{11}$$
$$p(y_i, z_i = 1|\boldsymbol{\theta}) = ab^{y_i}(1-b)^{1-y_i} \tag{12}$$

接下来计算 $p(z_i = 0|y_i, \boldsymbol{\theta}^t)$ 与 $p(z_i = 1|y_i, \boldsymbol{\theta}^t)$，由于有

$$P(Z|y_i, \boldsymbol{\theta}^t) = \frac{P(Z, y_i|\boldsymbol{\theta}^t)}{P(y_i|\boldsymbol{\theta}^t)} \tag{13}$$

和

$$P(y_i|\boldsymbol{\theta}^t) = (1-a^t)(c^t)^{y_i}(1-c^t)^{1-y_i} + a^t(b^t)^{y_i}(1-b^t)^{1-y_i} \tag{14}$$

故有

$$p(z_i = 0|y_i, \boldsymbol{\theta}^t) = \frac{(1-a^t)(c^t)^{y_i}(1-c^t)^{1-y_i}}{(1-a^t)(c^t)^{y_i}(1-c^t)^{1-y_i} + a^t(b^t)^{y_i}(1-b^t)^{1-y_i}} = 1 - \mu_i^t \tag{15}$$
$$p(z_i = 1|y_i, \boldsymbol{\theta}^t) = \frac{a^t(b^t)^{y_i}(1-b^t)^{1-y_i}}{(1-a^t)(c^t)^{y_i}(1-c^t)^{1-y_i} + a^t(b^t)^{y_i}(1-b^t)^{1-y_i}} = \mu_i^t \tag{16}$$

代入 (10) 式，有

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^t) &= \sum_{i=1}^{n}(p(z_i = 0|y_i, \boldsymbol{\theta}^t)ln(p(y_i, z_i = 0|\boldsymbol{\theta})) + p(z_i = 1|y_i, \boldsymbol{\theta}^t)ln(p(y_i, z_i = 1|\boldsymbol{\theta}))) \\ &= \sum_{i=1}^{n}((1-\mu_i^t)ln((1-a)c^{y_i}(1-c)^{1-y_i}) + \mu_i^t ln(ab^{y_i}(1-b)^{1-y_i})) \end{aligned} \tag{17}$$

得到了对数似然关于 Z 的期望，E 步完成。
M 步推导如下寻找参数最大化期望似然，可得到更新规则

$$\boldsymbol{\theta}^{t+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^t) \tag{18}$$

将 (17) 式分别对 a,b,c 求偏导即可，令偏导数等于零，可得

$$\frac{\partial Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^t)}{\partial a} = 0$$
$$a^{t+1} = \frac{1}{n}\sum_{i=1}^{n}\mu_i^t \tag{19}$$
$$\frac{\partial Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^t)}{\partial b} = 0$$
$$b^{t+1} = \frac{\sum_{i=1}^{n} y_i\mu_i^t}{\sum_{i=1}^{n}\mu_i^t} \tag{20}$$
$$\frac{\partial Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^t)}{\partial c} = 0$$
$$c^{t+1} = \frac{\sum_{i=1}^{n} y_i(1-\mu_i^t)}{\sum_{i=1}^{n}(1-\mu_i^t)} \tag{21}$$

此即参数更新规则，M 步完成。

(b) 对于 K-means 算法，其算法中估计的参数为均值向量 $\{\mu_1, \mu_2, ..., \mu_k\}$；计算样本与各均值向量的距离、根据距离确定样本的簇标记并将样本划入相应的簇，是算法的 E 步；计算新的均值向量并进行更新是 M 步。

若 σ 的值已知，并且 $\sigma \to 0$，所以此时 EM 算法只需确定 μ_i, π_i $(1 \le i \le k)$，由于方差 $\sigma \to 0$，所以每类样本的分布非常集中，无限趋近于该类样本的均值，所以在 E 步中可以直接指派离样本点最近的均值向量所在的成分给该样本而无需对每个混合成分计算概率（即直接选取均值向量离样本点最近的混合成分使其 $\gamma_{ji} = 1$，其他成分的 $\gamma_{ji} = 0$），故参数 π_i 也不需要估计了，在 M 步中只需更新 $\mu_i^{'} = \frac{\sum_{j=1}^{N} \gamma_{ji} x_j}{\sum_{j=1}^{N} \gamma_{ji}} = \frac{\sum_{x \in C_i} x}{|C_i|}$，故此时 EM 算法就相当于 K-means 聚类。不同之处在于 K-means 算法并不是对隐变量分布进行建模，也不考虑样本的所属类别也存在一个分布，而是将样本划入最近的均值所属的类别，直接对样本的类别进行硬指定，当 σ 不满足以上条件时，K-means 算法与 EM 算法存在不同。

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) 由于最大化目标为样本投影到主成分后的投影点之间的方差，故具体的优化目标可写为

$$\max_w \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - w^T \hat{x})^2 \tag{22}$$
$$s.t. ||w||^2 = 1$$

又可假设数据已经中心化，故有 $\hat{x}=0$，(22) 式可改写为如下式，此即为具体的优化目标。

$$\begin{aligned}&\max_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i)^2 \\ &s.t. ||w||^2 = 1\end{aligned} \tag{23}$$

对优化目标函数展开可得

$$\begin{aligned}&\max_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i)^2 \\ =&\max_w \frac{1}{n} w^T X^T X w\end{aligned}$$

又有约束条件 $||w||^2=1$，故有

$$\begin{aligned}&\max_w \frac{1}{n} w^T X^T X w \\ =&\max_w \frac{1}{n}\frac{w^T X^T X w}{||w||^2} \\ =&\max_w \frac{1}{n}\frac{w^T X^T X w}{w^T w}\end{aligned} \tag{24}$$

可以发现优化目标变成了瑞利商除以样本数 n，此即优化目标与瑞利商的关系。

(b)

方法与 (a) 问类似，只是原本的样本点 x_i 被替换为减去沿 (a) 中 w 方向的分量的 x_i（即只保留 x_i 中垂直 (a) 问中 w 方向的分量）。

定义 $x_i^{'}=x_i-w^T x_i w$，可证 $x_i^{'}$ 垂直 w

$$\begin{aligned}w^T x_i &= w^T(x_i - w^T x_i w) \\ &= w^T(x_i - w^T x_i w) \\ &= w^T x_i - w^T w^T x_i w\end{aligned} \tag{25}$$

又 w^T 为 d 维行向量，x_i 为 d 维列向量，故 $w^T x_i$ 为常数，(25) 式可继续化简为

$$\begin{aligned}w^T x_i^{'} &= w^T x_i - w^T w^T x_i w \\ &= w^T x_i - w^T x_i w^T w \\ &= w^T x_i - w^T x_i \\ &= 0\end{aligned}$$

即 $x_i^{'}$ 垂直 w。设第二个主成分为 v，则应有 $v^T w=0$，又 $w^T x_i w$ 与第一个主成分 w 平行，故有 $v^T w^T x_i w=0$，进一步有下式成立

$$\begin{aligned}v^T x_i &= v^T x_i - v^T w^T x_i w \\ &= v^T(x_i - w^T x_i w) \\ &= v^T x_i^{'}\end{aligned} \tag{26}$$

即 v 与 $x_i^{'}$ 正交即可保证与 x_i 正交，故选取第二个主成分只需对 $x_i^{'}=x_i-w^T x_i w$ 应用 (a) 问中的方法即可，故只需求解优化问题

$$\begin{aligned}&\max_w \frac{1}{n}\sum_{i=1}^{n}(w^T x_i^{'})^2 \\ &s.t. ||w||^2 = 1\end{aligned} \tag{27}$$

2. (a) 区别：PCA 只支持线性降维，而 AutoEncoder 支持线性和非线性降维。

   联系：AutoEncoder 可以看作基于神经网络的目标输出近似于输入的恒等函数，当 AutoEncoder 只有单隐层时，若将利用 PCA 降维的过程看作神经网络中不带有偏置的一层全连接映射，则基于最近重构性的 PCA 推导方法与 AutoEncoder 十分相似，都是基于神经网络使目标输出近似于输入，都可用于降维。

   (b) 常用的二范数正则化 $\|W\|_2^2$ 在这个问题中由于约束条件 $||w||_2^2 = 1$ 而不可用，零范数正则化 $\|W\|_0$ 不易求解，也不适用。故选择零范数正则化 $\|W\|_1$，加入正则化项后，优化目标可写为

$$\min_w -\frac{1}{n}\sum_{i=1}^{n}(w^T x_i)^2 + \lambda||w||_1 \tag{28}$$
$$s.t. ||w||_2^2 = 1$$

   L1 正则化的优化问题可使用 LARS 或坐标下降法求解。

3. (a) 了解了 sklearn 中实现的 SVD 的方法，主要对比一下 FullSVD、TruncatedSVD 和 RandomizedSVD。

   FullSVD：通过 scipy.linalg.SVD 调用 standard LAPACK solver 运行完全 SVD，返回完全分解后的结果，再通过后处理选择前 k 个主成分。

   TruncatedSVD：通过 scipy.sparse.linalg.svds 调用 ARPACK solver 运行 SVD 截断到第 k 个主成分即返回，不需要求解出所有奇异值。

   RandomizedSVD：使用随机算法求解 SVD，随机算法的原理大致如下：

   1. 使用随机采样方式构建低维矩阵 Q，使得 $A \approx QQ^T A$。

   2. 构建小矩阵 $B = Q^T A$，并对矩阵 B 进行奇异值分解，得到矩阵 A 的近似解。

   在原始空间维数较大时，RandomizedSVD 由于只需要在低维矩阵 B 上进行 SVD 分解，性能优势明显，运行效率最高；而对于另两种求解方法，由于 TruncatedSVD 不需要完全分解，故运行效率要高于 FullSVD；FullSVD 要对原始矩阵进行完全分解，运行效率最低。

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。最后两道任选一题即可。

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. (10 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. (10 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. 为简化式子，令 $d_{ij} = \exp\left(-\|Ax_i - Ax_j\|_2^2\right)$，则 p_{ij} 可写为 $p_{ij} = \frac{d_{ij}}{\sum_{k\neq i} d_{ik}}$

   又有

$$\begin{aligned}\frac{\partial d_{ij}}{\partial A} &= \frac{\partial \exp\left(-\|Ax_i - Ax_j\|_2^2\right)}{\partial A} \\ &= -2d_{ij}A(x_i - x_j)(x_i - x_j)^T\end{aligned} \tag{29}$$

   故有

$$\begin{aligned}\frac{\partial p_{ij}}{\partial A} &= \frac{\partial \frac{d_{ij}}{\sum_{k\neq i} d_{ik}}}{\partial A} \\ &= \frac{1}{\left(\sum_{k\neq i} d_{ik}\right)^2}\left(\frac{\partial d_{ij}}{\partial A}\sum_{k\neq i} d_{ik} - d_{ij}\sum_{k\neq i}\frac{\partial d_{ik}}{\partial A}\right) \\ &= -2p_{ij}A\left((x_i - x_j)(x_i - x_j)^T - \sum_{k\neq i} p_{ik}(x_i - x_k)(x_i - x_k)^T\right)\end{aligned} \tag{30}$$

   由 $f = \sum_{i=1}^n p_i = \sum_{i=1}^n \sum_{j\in C_i} p_{ij}$ 可得

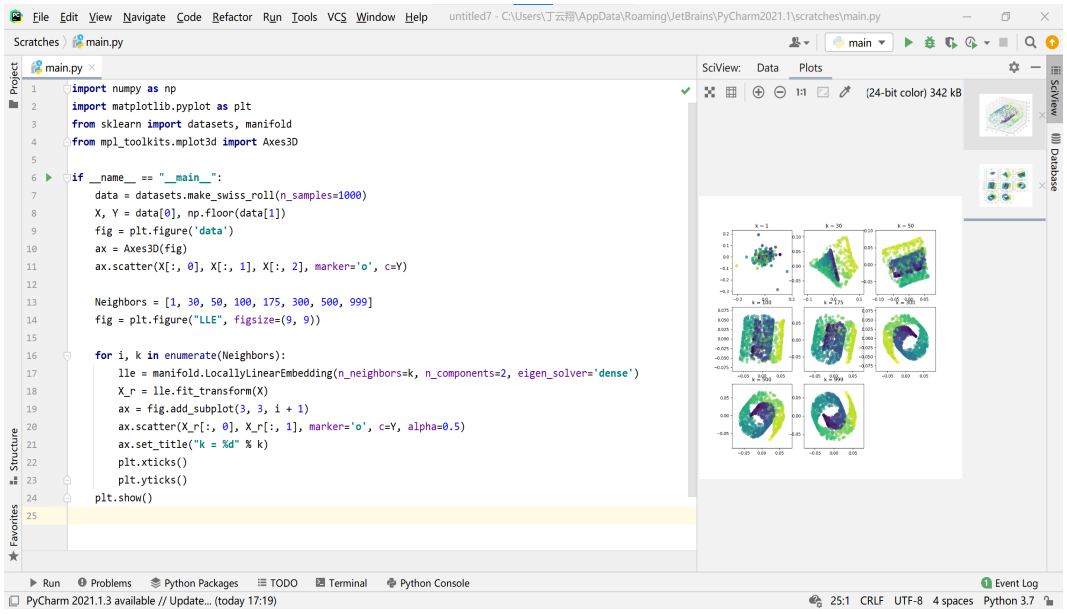
$$\begin{aligned}\frac{\partial f}{\partial A} &= \frac{\partial(\sum_{i=1}^n p_i)}{\partial A} \\ &= \frac{\partial(\sum_{i=1}^n \sum_{j\in C_i} p_{ij})}{\partial A} \\ &= \sum_{i=1}^n \sum_{j\in C_i}\frac{\partial p_{ij}}{\partial A} \\ &= -2A\sum_{i=1}^n \sum_{j\in C_i} p_{ij}\left((x_i - x_j)(x_i - x_j)^T - \sum_{k\neq i} p_{ik}(x_i - x_k)(x_i - x_k)^T\right) \\ &= 2A\sum_{i=1}^n\left(p_i\sum_{k\neq i} p_{ik}(x_i - x_k)(x_i - x_k)^T - \sum_{j\in C_i} p_{ij}(x_i - x_j)(x_i - x_j)^T\right)\end{aligned} \tag{31}$$

2. LSA 的具体步骤为：首先根据文档集合和词汇集合，统计出每个词汇在每个文档中出现的次数，并组织为矩阵的形式，然后对样本矩阵 $\boldsymbol{X}$ 进行 SVD 分解（$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{S}\boldsymbol{D}^T$，其中 $\boldsymbol{S}$ 为一个对角矩阵，对角线上的值为 $\boldsymbol{X}$ 的奇异值），然后保留矩阵的前 k 个奇异值，奇异向量也只取前 k 个奇异值对应的，得到 $\boldsymbol{S}^*, \boldsymbol{T}^*, \boldsymbol{D}^*$，则原矩阵 $\boldsymbol{X}$ 可以降维为 $\boldsymbol{X}^* = \boldsymbol{T}^*\boldsymbol{S}^*\boldsymbol{D}^{*T}$，然后可以直接在矩阵 $\boldsymbol{X}^*$ 上进行分析。

   与 PCA 的区别：是否需要中心化不同，LSA 进行 SVD 分解前一般不用中心化，而 PCA 则需要；进行 SVD 的矩阵不同，LSA 是对术语文档矩阵进行 SVD 分解，而 PCA 是对协方差矩阵进行 SVD 分解；分解得到的奇异向量含义也不相同，LSA 的奇异向量表示单词/文档在潜在语义空间上的坐标，而 PCA 的奇异向量表示主成分。

3. 直接使用 sklearn 库的 manifold.LocallyLinearEmbedding 来实现 LLE。数据集采用瑞士卷数据，使用 sklearn 库的 datasets.make_swiss_roll 来生成数据，共生成 1000 个样本点，将其降维到二维。LocallyLinearEmbedding 函数的参数主要有 n_neighbors（构建近邻的样本点个数，对实验结果影响较大，主要对比这一参数的影响）, n_components（降维到的维数，这里降维至二维）, eigen_solver（特征分解的算法，指定为 “dense” 以得到稳定解）, method（LLE 算法，不指定）, neighbors_algorithm（寻找 K 近邻的算法，不指定）。

实验代码和实验结果如下，可见当构建近邻的样本点个数提高到 300 后，降维效果较明显。



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, manifold
from mpl_toolkits.mplot3d import Axes3D

if __name__ == "__main__":
    data = datasets.make_swiss_roll(n_samples=1000)
    X, Y = data[0], np.floor(data[1])
    fig = plt.figure('data')
    ax = Axes3D(fig)
    ax.scatter(X[:, 0], X[:, 1], X[:, 2], marker='o', c=Y)

    Neighbors = [1, 30, 50, 100, 175, 300, 500, 999]
    fig = plt.figure("LLE", figsize=(9, 9))

    for i, k in enumerate(Neighbors):
        lle = manifold.LocallyLinearEmbedding(n_neighbors=k, n_components=2, eigen_solver='dense')
        X_r = lle.fit_transform(X)
        ax = fig.add_subplot(3, 3, i + 1)
        ax.scatter(X_r[:, 0], X_r[:, 1], marker='o', c=Y, alpha=0.5)
        ax.set_title("k = %d" % k)
        plt.xticks()
        plt.yticks()
    plt.show()
```

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：

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,nh}^j\right)^2 + \operatorname{diff}\left(x_i^j, x_{i,\mathrm{nm}}^j\right)^2 \tag{32}$$

多分类的 Relief-F 算法的相关统计量计算公式如下：

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,\mathrm{nh}}^j\right)^2 + \sum_{l\neq k}\left(p_l \times \operatorname{diff}\left(x_i^j, x_{i,l,\mathrm{nm}}^j\right)^2\right) \tag{33}$$

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：

由 $\sum_{l\neq k} p_l = 1 - p_k < 1$ 可得 $\sum_{l\neq k} \frac{p_l}{1-p_k} = 1$。

故将系数 p_l 改为 $\frac{p_l}{1-p_k}$ 即可，修正后的计算公式如下

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,\mathrm{nh}}^j\right)^2 + \sum_{l\neq k}\left(\frac{p_l}{1-p_k} \times \operatorname{diff}\left(x_i^j, x_{i,l,\mathrm{nm}}^j\right)^2\right) \tag{34}$$

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

$$\begin{array}{ll}\max\limits_{\theta} & \left(\nabla L_{\theta_{\text{old}}}(\theta)\right)^{T}\left(\theta-\theta_{\text{old}}\right) \\ \text{s.t.} & \frac{1}{2}\left(\theta-\theta_{\text{old}}\right)^{T} H\left(\theta-\theta_{\text{old}}\right) \leq \delta\end{array} \tag{35}$$

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 $L_{\theta_{\text{old}}}$ 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

$$\begin{array}{ll}\max_{w} & \max_{\theta}\left(\nabla\left(L^{T} w\right)\right)^{T}\left(\theta-\theta_{\text{old}}\right) \\ \text{s.t.} & \frac{1}{2}\left(\theta-\theta_{\text{old}}\right)^{T} H\left(\theta-\theta_{\text{old}}\right) \leq \delta \\ & \|w\|_1=1 \\ & w_i \geq 0, \quad i=1,2, \ldots, n\end{array} \tag{36}$$

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. 现实意义在于: 原目标 L 现由多个子目标组成, 每个子目标都有一个权重 w_i, 所有子目标合起来应该还原成一个原目标, 原目标的权重为 1, 故 w 应满足 L1 范数约束 $\|w\|_1=\sum_{i=1}^{n} w_i=1$
2. 需要注意的事项有: 特征选择可以利用已知的先验知识, 来缩小特征选择范围, 避免不必要的探索, 提高特征效率。可以引入塑形权重函数来表示权重。