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) 我们可以采用“再缩放”的思想，具体做法可以有如下几种：
   一、欠采样：减少训练集中的多数类样本使得各个类别样本数目相近，再进行学习，如 EasyEnsemble 和 BalanceCascade 算法。
   二、过采样：即增加一些少数类样本使得各个类别样本数目接近，然后再进行学习，如 SMOTE 算法等。
   三、阈值移动：在数据不平衡时，默认的阈值会导致模型输出倾向于类别数据多的类别，阈值移动通过改变决策阈值来偏重少数类。比如线性分类器 $y = w^T x + b$ 就是用计算出的 y 与阈值 (默认 0.5) 比较，大于则分为正例，反之则分为反例。为了避免类别不平衡的影响，我们对决策规则做如下修正：

$$\frac{y}{1-y} > 1, \text{预测为正} \Rightarrow \frac{y}{1-y} * \frac{m^-}{m^+} > 1, \text{预测为正}$$

   其中 m^+, m^- 分别是样本中正例，反例的数目
   此外，对于上式中的 m^-/m^+，若用 $cost^+/cost^-$(正例误分为反例的代价/反例误分为正例的代价) 代替，就是代价敏感学习。

   (b) 优化问题为：

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

   对偶问题的推导：
   先写出拉格朗日函数

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\zeta}, \boldsymbol{\mu}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{m} p_i \zeta_i + \sum_{i=1}^{m} \alpha_i (1 - \zeta_i - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) - \sum_{i=1}^{m} \mu_i \zeta_i$$

其中 $\alpha_i \geq 0,\ \mu_i \geq 0$ 是拉格朗日乘子
接下来分别对 $\boldsymbol{w}, b, \zeta_i$ 求偏导并令其 $= 0$ 可得

$$\begin{cases} \boldsymbol{w} = \sum\limits_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \\ 0 = \sum\limits_{i=1}^{m} \alpha_i y_i \\ Cp_i = \alpha_i + \mu_i \end{cases}$$

将这三个式子代入拉格朗日函数即可得到对偶问题:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ s.t. \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq Cp_i,\ i = 1, 2, ..., m \end{aligned}$$

2. (a) 记参数 $\theta = (a, b, c)$，硬币 A 的投掷结果无法观测，为隐变量 $Z = (Z_1, Z_2, ..., Z_n)^T$，第二次投掷的硬币结果为可观测变量 $Y = (Y_1, ..., Y_n)^T$
在一次投掷中，得到结果 $y = 0, 1$ 的概率 $P(y|\theta) = ab^y(1-b)^{1-y} + (1-a)c^y(1-c)^{1-y}$
我们可以写出似然函数

$$P(Y|\theta) = \prod_{j=1}^{n} [ab^{y_j}(1-b)^{1-y_j} + (1-a)c^{y_j}(1-c)^{1-y_j}]$$

需要求解参数的最大似然估计

$$\hat{\theta} = arg\max_{\theta} logP(Y|\theta)$$

首先对参数进行初始化 (随机初始化即可)，得到 $\theta^{(0)} = (a^{(0)}, b^{(0)}, c^{(0)})$，接下来按照如下步骤迭代，直到收敛或者达到迭代次数为止，记第 i 次迭代时的参数为 $\theta^{(i)} = (a^{(i)}, b^{(i)}, c^{(i)})$，第 i+1 次迭代如下:
E 步：计算在参数 $\theta^{(i)}$ 下观测到的数据 y_j 来自硬币 B 的概率

$$\mu_j^{(i+1)} = \frac{a^{(i)}(b^{(i)})^{y_j}(1-b^{(i)})^{1-y_j}}{a^{(i)}(b^{(i)})^{y_j}(1-b^{(i)})^{1-y_j} + (1-a^{(i)})(c^{(i)})^{y_j}(1-c^{(i)})^{1-y_j}}$$

M 步：对参数值进行更新

$$\begin{aligned} a^{(i+1)} &= \frac{1}{n} \sum_{j=1}^{n} \mu_j^{i+1} \\ b^{(i+1)} &= \frac{\sum\limits_{j=1}^{n} \mu_j^{(i+1)} y_j}{\sum\limits_{j=1}^{n} \mu_j^{(i+1)}} \\ c^{(i+1)} &= \frac{\sum\limits_{j=1}^{n} (1 - \mu_j^{(i+1)} y_j)}{\sum\limits_{j=1}^{n} (1 - \mu_j^{(i+1)})} \end{aligned}$$

(b)K-means 在一次迭代中的两步，可以分别看作是 EM 算法的 E 步和 M 步。
其中，把各样本点依据到簇中心的距离分到最近的簇对应于 E 步
更新各簇的均值向量作为新的簇中心对应于 M 步
区别：K-Means 算法中每次对参数的更新是硬猜测，而 EM 中每次对参数的更新是软猜测

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)
我们的目标是最大化方差

$$Var(w^T x_i) = \frac{1}{m}\sum_{i=1}^{m}(w^T x_i - \frac{1}{m}(\sum_{i=1}^{m} w^T x_i))^2$$

考虑到样本已经中心化，即均值为零，忽略系数后我们要最大化 $\sum_{i=1}^{m}(w^T x_i)^2$
此时优化问题为：

$$\max_w \sum_{i=1}^{m}(w^T x_i)^2 = w^T X^T X w$$
$$s.t. \quad w^T w = 1$$

再做整理可得

$$w = \arg\max \frac{w^T X^T X w}{w^T w}$$

等式右边即为瑞利商，最大值即为 $X^T X$ 的最大的特征值，此时 w 为对应的单位特征向量

(b)
对于向量 x，其与 w 平行的分量为 $w^T x_i * w$，故对所有 x_i 减去与 w 平行的部分，得到 $x_i' = x_i - w^T x_i * w$
(分解方法为 $x_i = w^T x_i * w$(平行于 w) + $(x_i - w^T x_i * w)$(垂直于 w))
此时对应的数据矩阵 $X' = X - X w w^T$，该矩阵中保留了全部垂直于 w 的向量，故对该部分数据选取第二主成分。
我们有

$$v = \arg\max \frac{v^T (X')^T X' v}{v^T v}$$

可以求得 v 是 $(X')^T X'$ 的最大特征值对应的单位特征向量，即 $X^T X$ 的第二大特征值对应的单位特征向量。

2. (a)
(1)、这二者都可以用来降低数据维数，但是 PCA 无法用于学习非线性特征，AutoEncoder 可以。
(2)、PCA 在降维后，必然存在信息损失，但是 Autoencoder 降维的方法是对数据进行数据编码再进行解码，之后最小化原数据与解码数据之间的误差从而学习到变换矩阵，这种方法虽然损失很小，但要付出更多的计算量。
(3)、从公式上看，基于最近重构性的 PCA 要最小化函数

$$\sum_{i=1}^{m} || \sum_{j=1}^{d'} z_{ij} \boldsymbol{w}_j - \boldsymbol{x}_j ||_2^2$$

其中，d' 为降维后的空间的维数，$\boldsymbol{w_j}$ 是低维空间的第 j 个标准正交基向量，z_{ij} 为原样本点投影到新坐标系的相应系数。
而 Autoencoder 则是最小化

$$\sum_{i=1}^{m} ||x_i' - x_i||_p^p$$

其中 x_i 是原数据，x_i' 是编码后再解码的数据，求解时通常基于反向传播算法。
(b) 优化问题如下：

$$\arg\min_{A,B} \sum_{i=1}^{n} ||x_i - A B^T x_i||_2^2 + \lambda_1 \sum_{j=1}^{k} ||\beta_j||_2^2 + \lambda_2 \sum_{j=1}^{k} ||\beta_j||_1$$
$$s.t. \qquad A^T A = I^{k*k}$$

其中 $A = [\alpha_1, \alpha_2, ..., \alpha_k]$ 为前 k 个非稀疏主成分，$B = [\beta_1, ..., \beta_k]$ 为前 k 个稀疏主成分。
求解方法大致为：
1、固定 A，利用 L_1 正则化求解线性回归问题从而得到 B
2、固定 B，利用 $X^T X B = U \Sigma V^T, A^* = U V^T$ 求解最优解 A^*。

3. (a) 区别如下：
FULLSVD 是直接计算 $m*n$ 矩阵 A 的奇异值分解 ($A = U\Sigma V^T$)，返回 U, s, V^T 的结果 (Σ 作为列向量 s 返回)，并且对 s 中的奇异值从大到小排序，fullsdiag() 可以将列向量 s 转化为奇异值分解中的矩阵 Σ

TruncatedSVD 是实现 SVD 的一种变体，通过截断奇异值分解进行降维，它只计算 k(由用户指定) 个最大奇异值，即 $X \approx X_k = U_k\Sigma_k V_k^T$，当应用于单词-文本矩阵时，这种变换即为潜在语义分析 (LSA)。并且，这种变换适用于任何特征矩阵。

RandomizedSVD 用于计算截断的随机 SVD，通过随机化来找到（通常表现非常好的）近似截断奇异值分解来加速计算，尤其是在只保留少量特征的大型矩阵上有着优异的表现。
下表是对于一个随机生成的 $1500*1500$ 矩阵，三种方法作奇异值分解各自消耗的时间 (截断奇异值分解取 k=300，表现最佳的数据用红色表示)：

实验次数	FullSVD	TruncatedSVD	RandomizedSVD
1	2811.2144ms	1091.3017ms	838.4232ms
2	2269.7217ms	888.2902ms	633.3055ms
3	2992.6169ms	1138.2227ms	632.8134ms
4	3148.6489ms	1132.6401ms	666.2230ms
5	3089.3362ms	1359.3519ms	812.2656ms

实验体现出来的运行性能和三种 SVD 的描述相对应的情况还是很吻合的，FullSVD 最慢，截断奇异值分解次之，随机截断奇异值分解最快。

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，$p_{ij} = \frac{\exp(-\|Ax_i - Ax_j\|_2^2)}{\sum_{k \neq i} \exp(-\|Ax_i - Ax_k\|_2^2)}, j \neq i, p_{ii} = 0$ 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. 方便起见记 $g_{ij} = exp(-||Ax_i - Ax_j||_2^2)$，此时 $p_{ij} = \frac{g_{ij}}{\sum_{k \neq i} g_{ik}}$
根据链式法则，一步一步求导

$$\frac{\partial p_{ij}}{\partial A} = \frac{1}{(\sum_{k \neq i} g_{ik})^2} (\frac{\partial g_{ij}}{\partial A} \sum_{k \neq i} g_{ik} - g_{ij} \sum_{k \neq i} \frac{\partial g_{ik}}{\partial A}) \quad (1)$$

再求 g_{ij} 对 A 的偏导

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

然后再求 f 对 A 的偏导

$$\frac{\partial f}{\partial A} = \sum_{i=1}^{n} \sum_{j \in C_i} \frac{\partial p_{ij}}{\partial A}$$

将 (2) 式代入 (1) 消去 g_{ij} 后再把 p_{ij} 的偏导代入梯度表达式，即可得到要求的梯度。

此处省略化简过程，直接给出计算的结果

$$\frac{\partial p_{ij}}{\partial A} = -2p_{ij}A[(x_i - x_j)(x_i - x_j)^T - \sum_{k \neq i} p_{ik}(x_i - x_k)(x_i - x_k)^T]$$

$$\frac{\partial f}{\partial A} = 2A\sum_{i}[p_i \sum_{k \neq i} p_{ik}(x_i - x_k)(x_i - x_k)^T - \sum_{j \in C_i} p_{ij}(x_i - x_j)(x_i - x_j)^T]$$

2. 首先，要介绍话题向量空间，对于单词-文本矩阵 X，他构成原始的单词向量空间，每一列是一个文本在单词向量空间中的表示。假设所有文本共包含了 k 个话题，每个话题由一个定义在单词集合 W 上的 m 维向量表示 (即话题向量)，那么这 k 个话题向量就张成了一个话题向量空间。

潜在语义分析算法，就是对单词-文本矩阵进行奇异值分解，将其左矩阵作为话题向量空间，对角矩阵和右矩阵的乘积作为文本在话题向量空间中的表示。其步骤大致如下：
(1)、给定文本集合 $D = \{d_1, ..., d_n\}$ 和单词集合 $W = \{w_1, ...w_m\}$，将其表示成一个单词-文本矩阵：

$$X = \begin{bmatrix} x_{11} & x_{12} & ... & x_{1n} \\ x_{21} & x_{22} & ... & x_{2n} \\ ... & ... & ... & ... \\ x_{m1} & x_{m2} & ... & x_{mn} \end{bmatrix}$$

其中，x_{ij} 表示单词 w_i 在文本 d_j 中出现的权值
(2)、根据给定的话题数 k 对单词-文本矩阵 X 进行截断奇异值分解

$$X \approx U_k \Sigma_k V_k^T = [u_1 ... u_k] \begin{bmatrix} \sigma_1 & 0 & 0 & ... \\ 0 & \sigma_2 & 0 & ... \\ & & \ddots & \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & ... & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$$

在该式子中，$k \leq n \leq m$，U_K 是 $m * k$ 矩阵，它的列由 X 的前 k 个互相正交的左奇异向量组成，Σ_k 是 k 阶对角方阵，对角元素为前 k 个最大奇异值，V_k 是 $n * k$ 矩阵，它的列由 X 的前 k 个互相正交的右奇异向量组成。

U_k 中的每一个列向量表示一个话题 (即话题向量)，这 k 个话题向量张成一个子空间 U_k 就称为话题向量空间。

接下来考虑文本在话题空间的表示，由截断奇异值分解：$X \approx U_k \Sigma_k V_k^T$，故 X 的第 j 列向量 x_j 满足

$$x_j \approx U_k(\Sigma_k V_k^T)_j = \sum_{l=j}^{k} \sigma_l v_{jl} u_l,\ j = 1, ..., n$$

其中 $(\Sigma_k V_k^T)_j$ 是矩阵 $(\Sigma_k V_k^T)$ 的第 j 列向量，上式是文本 d_j 的近似表达式，由 k 个话题向量的线性组合表示。对于矩阵 $(\Sigma_k V_k^T)$，其每一个列向量都是一个文本在话题向量空间的表示

LSA 与 PCA 的区别:
(1)、LSA 本质上做的就是奇异值分解，只是对分解后的各个矩阵赋予了实际意义的解释；而 PCA 中，奇异值分解只是一种求解方法，并没有涉及到其本质。

(2)、LSA 是寻找 F 范数中的最佳线性子空间，PCA 则是寻找最佳仿射线性子空间。

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:

将 p_l 替换为 $\frac{p_l}{1-p_k}$，此时多分类问题后一项的系数为 $\sum\limits_{l\neq k}\frac{p_l}{1-p_k} = \frac{\sum\limits_{l\neq k} p_l}{1-p_k} = \frac{1-p_k}{1-p_k} = 1$，修正完成。

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

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

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

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

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. 加入 w 后，我们对每个 L_i 就赋予了一个权重，通过调整不同质量的子目标的权重，可以使得优化更具有针对性，效果也更好。
   使用 w 的 L_1 范数进行约束，计算上更加简便，同时也更便于理解 w 的意义。
2. 在特征选择中，我们需要注意保留特征的数目，即不能过多，又不能过少。

   选取的特征过多时，可能仍然存在一部分冗余特征，不仅不能有效降低训练模型的开销，对模型的准确度也有负面影响；此外，由于去除的特征数量少，也无法完全规避过拟合的风险。

   选取的特征过少时，很容易出现欠拟合问题，同样不利于提升模型的准确度。