

姓名：方盛俊  
学号：201300035

# 一. (20 points) 神经网络基础

给定训练集  $D = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$ . 其中  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $\mathbf{y}_i \in \mathbb{R}^l$  表示输入示例由  $d$  个属性描述, 输出  $l$  维实值向量. 图 ?? 给出了一个有  $d$  个输入神经元、 $l$  个输出神经元、 $q$  个隐层神经元的多层神经网络, 其中输出层第  $j$  个神经元的阈值用  $\theta_j$  表示, 隐层第  $h$  个神经元的阈值用  $\gamma_h$  表示. 输入层第  $i$  个神经元与隐层第  $h$  个神经元之间的连接权为  $v_{ih}$ , 隐层第  $h$  个神经元与输出层第  $j$  个神经元之间的连接权为  $w_{hj}$ . 记隐层第  $h$  个神经元接收到的输入为  $\alpha_h = \sum_{i=1}^d v_{ih}x_i$ , 输出层第  $j$  个神经元接收到的输入为  $\beta_j = \sum_{h=1}^q w_{hj}b_h$ , 其中  $b_h$  为隐层第  $h$  个神经元的输出.

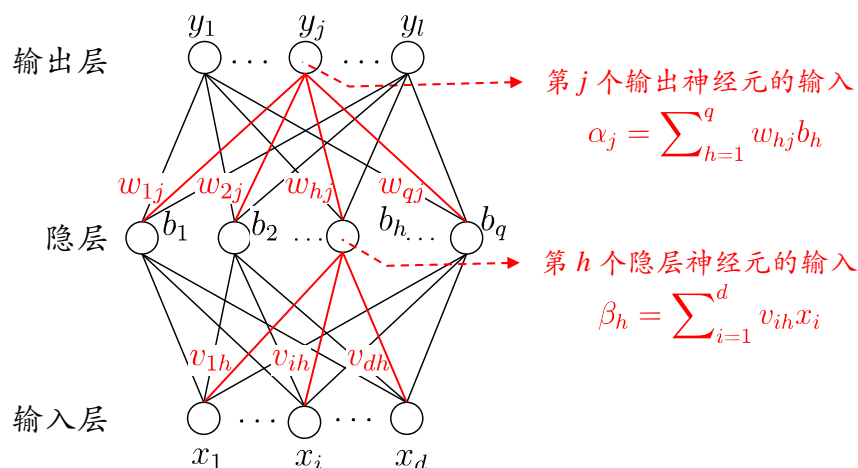


Figure 1: 多层神经网络 (教材图 5.7)

不同任务中神经网络的输出层往往使用不同的激活函数和损失函数, 本题介绍几种常见的激活和损失函数, 并对其梯度进行推导.

1. 在二分类问题中 ( $l = 1$ ), 标记  $y \in \{0, 1\}$ , 一般使用 Sigmoid 函数作为激活函数, 使输出值在  $[0, 1]$  范围内, 使模型预测结果可直接作为概率输出. Sigmoid 函数的输出一般配合二元交叉熵 (Binary Cross-Entropy) 损失函数使用, 对于一个训练样本  $(\mathbf{x}, y)$  有

$$\ell(y, \hat{y}_1) = -[y \log(\hat{y}_1) + (1 - y) \log(1 - \hat{y}_1)] \quad (1)$$

记  $\hat{y}_1$  为模型对样本属于正类的预测结果, 请计算  $\frac{\partial \ell(y, \hat{y}_1)}{\partial \beta_1}$ ,

2. 当  $l > 1$ , 网络的预测结果为  $\hat{\mathbf{y}} \in \mathbb{R}^l$ , 其中  $\hat{y}_i$  表示输入被预测为第  $i$  类的概率. 对于第  $i$  类的样本, 其标记  $\mathbf{y} \in \{0, 1\}^l$ , 有  $y_i = 1$ ,  $y_j = 0, j \neq i$ . 对于一个训练样本  $(\mathbf{x}, \mathbf{y})$ , 交叉熵损失函数  $\ell(\mathbf{y}, \hat{\mathbf{y}})$  的定义如下

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{j=1}^l y_j \log \hat{y}_j \quad (2)$$

在多分类问题中, 一般使用 Softmax 层作为输出, Softmax 层的计算公式如下

$$\hat{y}_j = \frac{e^{\beta_j}}{\sum_{k=1}^l e^{\beta_k}} \quad (3)$$

易见 Softmax 函数输出的  $\hat{\mathbf{y}}$  符合  $\sum_{j=1}^l \hat{y}_j = 1$ , 所以可以直接作为每个类别的概率. Softmax 输出一般配合交叉熵 (Cross Entropy) 损失函数使用, 请计算  $\frac{\partial \ell(\mathbf{y}, \hat{\mathbf{y}})}{\partial \beta_j}$ ,

3. 分析在二分类中使用 Softmax 和 Sigmoid 的联系与区别.  
4. KL 散度 (Kullback-Leibler divergence) 定义了两个分布之间的距离, 对于两个离散分布  $Q(x)$  和  $P(x)$ , 其定义为

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right) \quad (4)$$

其中  $\mathcal{X}$  为  $x$  的取值空间. 试分析交叉熵损失函数和 KL 散度的关系.

**解:**

1.

$$\begin{aligned} \frac{\partial \ell(y, \hat{y}_1)}{\partial \beta_1} &= \frac{\partial \ell(y, \hat{y}_1)}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial \beta_1} \\ &= \left( \frac{1-y}{1-\hat{y}_1} - \frac{y}{\hat{y}_1} \right) \cdot f'(\beta_1 - \theta_1) \\ &= \hat{y}_1(1-\hat{y}_1) \left( \frac{1-y}{1-\hat{y}_1} - \frac{y}{\hat{y}_1} \right) \\ &= \hat{y}_1 - y \end{aligned}$$

2. 令  $f(x) = \frac{e^x}{e^x + \theta}$ , 对其求导得  $f'(x) = \frac{e^x}{\theta + e^x} (1 - \frac{e^x}{\theta + e^x}) = f(x)(1 - f(x))$

令  $f(x) = \frac{\varphi}{e^x + \theta}$ , 对其求导得  $f'(x) = -\frac{\varphi}{\theta + e^x} (1 - \frac{\theta}{\theta + e^x}) = -f(x)(1 - \frac{\theta}{\varphi} f(x))$

$$\begin{aligned} \frac{\partial \ell(\mathbf{y}, \hat{\mathbf{y}})}{\partial \beta_j} &= \frac{\partial \ell(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial \beta_j} + \sum_{k \neq j} \frac{\partial \ell(\mathbf{y}, \hat{\mathbf{y}})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial \beta_j} \\ &= -\frac{y_j}{\hat{y}_j} \cdot \hat{y}_j (1 - \hat{y}_j) + \sum_{k \neq j} \frac{y_k}{\hat{y}_k} \cdot \hat{y}_k (1 - \frac{\sum_{i \neq j} e^{\beta_i}}{e^{\beta_k}} \hat{y}_k) \\ &= -y_j (1 - \hat{y}_j) + \sum_{k \neq j} y_k (1 - \frac{\sum_{i \neq j} e^{\beta_i}}{e^{\beta_k}} \hat{y}_k) \end{aligned}$$

3. 二分类, 即  $l = 1$ .

对于 (2) 中的结果带入  $j = 1$  即有

$$\begin{aligned} \frac{\partial \ell(\mathbf{y}, \hat{\mathbf{y}})}{\partial \beta_1} &= y_1 (\hat{y}_1 - 1) + y_0 (1 - \frac{\beta_0}{\beta_0} \hat{y}_0) \\ &= y_1 (\hat{y}_1 - 1) + (1 - y_1) \hat{y}_1 \\ &= \hat{y}_1 - y_1 \end{aligned}$$

和 (1) 中的结果相同, 因为 (1) 中的  $y$  即为  $y_1$ , 他们的计算结果均为  $\hat{y}_1 - y_1$ .

而 Sigmoid 函数  $\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$ ,

也与 Softmax 函数  $\text{softmax}(x_1, x_2) = \frac{e^{x_1}}{e^{x_1} + e^{x_2}} = \frac{1}{1 + e^{-(x_1 - x_2)}}$  完全相同 (在  $x = x_1 - x_2$  的定义的基础上).

虽然说在理论上对于二分类问题 Sigmoid 函数与 Softmax 函数完全相同, 但是对于建模之后的实际模型还是有所不同的.

Softmax 对两个类别均输出对应的概率, 并且两个类别概率的和为 1; Sigmoid 只输出一个类别的概率, 另一个类别使用 1 减去前

一类别的概率取得, 因此两个类别概率和仍为 1. 但是 Softmax 输出两类的概率, 而 Sigmoid 只输出一个类别的概率. 因此在实际模型构建和计算中, 肯定也还是会有所不同.

4.

$$\begin{aligned} D_{\text{KL}}(P||Q) &= \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \\ &= \sum_{x \in \mathcal{X}} P(x) \log P(x) - \sum_{x \in \mathcal{X}} P(x) \log Q(x) \\ &= b - \sum_{x \in \mathcal{X}} P(x) \log Q(x) \\ &= b + \ell(\mathbf{y}, \hat{\mathbf{y}}) \end{aligned}$$

其中  $x$  为第几个类别 (也即是  $j$ ),  $\mathcal{X}$  是类别的取值空间,  $P(x) = P(j) = y_j, Q(x) = Q(j) = \hat{y}_j$ ,

则我们有  $D_{\text{KL}}(P||Q) = b + \ell(\mathbf{y}, \hat{\mathbf{y}})$ , 其中  $b$  只与  $P(x)$  有关, 因此在  $P(x)$  即  $y_j$  分布不变的情况下可以视为一个常数.

所以我们可知,  $D_{\text{KL}}(P||Q)$  与  $\ell(\mathbf{y}, \hat{\mathbf{y}})$  只在所加的常数上有所区别. 我们最小化  $D_{\text{KL}}(P||Q)$  等价于最小化  $\ell(\mathbf{y}, \hat{\mathbf{y}})$ .

## 二. (20 points) 运算的向量化

在编程实践中, 一般需要将运算写成向量或者矩阵运算的形式, 这叫做运算的向量化 (vectorization). 向量化可以充分利用计算机体系结构对矩阵运算的支持加速计算, 大部分数学运算库例如 `numpy` 也对矩阵计算有专门的优化. 另一方面, 如果一个运算可以写成向量计算的形式, 会更容易写出其导数形式并进行优化. 本题中举两个简单的例子

1. 给定示例矩阵  $\mathbf{X} \in \mathbb{R}^{m \times d}$ , 表示  $m$  个示例 (向量), 每个示例有  $d$  维, 计算  $m$  个示例两两之间的距离矩阵  $\mathbf{D} \in \mathbb{R}^{m \times m}$ , 两个向量之间的欧式距离定义为  $\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$ . 求距离矩阵可以通过循环的方式, 即 `plain_distance_function` 中实现的方法;

```
1 import numpy as np
2
3 def plain_distance_function(X):
4     # 直观的距离计算实现方法
5     # 首先初始化一个空的距离矩阵D
6     D = np.zeros((X.shape[0], X.shape[0]))
7     # 循环遍历每一个样本对
```

```

8     for i in range(X.shape[0]):
9         for j in range(X.shape[0]):
10            # 计算样本i和样本j的距离
11            D[i, j] = np.sqrt(np.sum((X[i] - X[j])**2))
12     return D

```

2. 输入一个矩阵  $\mathbf{X} \in \mathbb{R}^{m \times d}$ , 表示  $m$  个向量, 每个向量有  $d$  维, 要求对输入矩阵的行按照一个给定的排列  $\mathbf{p} = \{p_1, p_2, \dots, p_m\}$  进行重新排列. 即输出一个新的矩阵  $\mathbf{X}'$ , 其中第  $i$  行的内容为输入矩阵的第  $p_i$  行. 假设重排列为一个函数 perm 即  $\mathbf{X}' = \text{perm}(\mathbf{X})$ , 已知梯度  $\frac{\partial \ell}{\partial \mathbf{X}'}$ , 需要计算  $\frac{\partial \ell}{\partial \mathbf{X}}$ . 对矩阵的行进行排列可以采用简单的循环实现, 例如 plain\_permutation\_function 中的实现方法.

```

1 import numpy as np
2
3 def plain_permutation_function(X, p):
4     # 初始化结果矩阵, 其中每一行对应一个样本
5     permuted_X = np.zeros_like(X)
6     for i in range(X.shape[0]):
7         # 采用循环的方式对每一个样本进行重排列
8         permuted_X[i] = X[p[i]]
9     return permuted_X

```

请给出上述两种任务的向量化实现方案, 并分析上述实现方法和向量化实现方法之间运行时间的差异。(提示: 比如可以针对不同规模的矩阵大小来尝试分析主要操作的运行时间)

解:

1. 使用 numpy 可以有两种方式将向量距离矩阵计算出来.

一种是使用 numpy 的张量功能 (这里即为 3 阶张量, 即三维数组), 通过生成新的维度即可保证不互相冲突. 然后一句简洁  
`"np.sqrt(((X[:, np.newaxis, :] - X[np.newaxis, :, :])**2).sum(axis=-1))"`

即可计算出结果. 我们记这种方法为 "tensor\_distance\_function".

另一种是将  $\sum_{i=1}^d (x_i - y_i)^2$  拆分成  $\sum_{i=1}^d x_i^2 + \sum_{i=1}^d y_i^2 - 2 \sum_{i=1}^d x_i y_i$ , 即拆分成三个不同的矩阵分别计算.

我们记这种方法为 "matrix\_distance\_function".

为了计算第一个矩阵  $\sum_{i=1}^d x_i^2$ , 即  $\mathbf{X}$  的所有行向量逐元素平方后与  $\mathbf{1}$  向量点乘. 我们首先要定义一个  $d \times m$  的全一矩阵  $\mathbf{1}$ , 然后

就可以通过  $\mathbf{M}_1 = \text{square}(\mathbf{X}) \cdot \mathbf{1}$  计算出第一个矩阵; 第二个矩阵  $\sum_{i=1}^d y_i^2$  可以通过  $\mathbf{M}_1$  转置生成, 即  $\mathbf{M}_2 = \mathbf{M}_1^T$ ; 而  $\sum_{i=1}^d x_i y_i$  矩阵则更为简单, 通过点乘即  $\mathbf{M}_3 = \mathbf{X} \cdot \mathbf{X}^T$  即可计算出第三个矩阵.

最后对这三个矩阵加权求和并开方即可算出最后的矩阵, 即  $\mathbf{M} = \text{sqrt}(\mathbf{M}_1 + \mathbf{M}_2 - 2\mathbf{M}_3)$ .

不过这个式子理论上正确, 实际上却会出问题. 对角线元素本来应该计算为 0, 不过由于计算的误差, 计算结果很有可能生成一个非常小的负数, 而负数是无法开方的, 导致生成 NaN 而计算结果出错. 所以我们需要在原来式子基础上加上一个很小的正数, 最终为  $\mathbf{M} = \text{sqrt}(\mathbf{M}_1 + \mathbf{M}_2 - 2\mathbf{M}_3 + 10^{-10} \cdot \mathbf{1})$

最终计算结果如下 ( $m$  和  $d$  为矩阵维度,  $n$  代表循环次数):

size	plain	tensor	matrix
$m = 10, d = 10, n = 10000$	16.97 s	0.23 s	0.39 s
$m = 1000, d = 1000, n = 10$	179.39 s	186.22 s	0.93 s

可以看出, 对于维度小的矩阵, 使用张量法和矩阵法效率差不多, 而且运行时间是朴素方法的 1/100 级别; 对于维度大的矩阵, 使用矩阵法的运行时间 0.93 秒远远小于张量法和朴素法的超过 100 秒.

因此, 对于这个问题, 使用矩阵法所取得的效果十分显著.

2. 我们可以对于任何一个给定的排列  $\mathbf{p} = \{p_1, p_2, \dots, p_m\}$ , 生成一个  $m \times m$  permutation 矩阵  $\mathbf{P}$ . 其中  $\mathbf{P}$  矩阵的生成方式为, 对于矩阵  $\mathbf{P}$  第  $i$  行, 在第  $p_i$  列的位置置 1, 其余均为 0. 则  $\mathbf{X}' = \mathbf{P} \cdot \mathbf{X}$  则为重新排列行向量的矩阵.

最终计算结果如下 ( $m$  和  $d$  为矩阵维度,  $n$  代表循环次数):

size	plain	matrix
$m = 10, d = 10, n = 100000$	3.90 s	2.18 s
$m = 1000, d = 1000, n = 1000$	12.69 s	63.58 s

可以看出, 对于维度小的矩阵, 使用朴素法和矩阵法效率差不多, 矩阵法最多比朴素法快了 1/3; 但是对于维度大的矩阵就恰好相反了, 因为矩阵法每次都需要生成巨大的 permutation 矩阵  $\mathbf{P}$ , 最后总耗时反而是朴素法的 6 倍.

### 三. (20 points) 支持向量机

考虑标准的 SVM 优化问题如下 (即教材公式 (6.35)),

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i \in [m]. \end{aligned} \tag{5}$$

注意到, 在 (2.1) 中, 对于正例和负例, 其在目标函数中分类错误的“惩罚”是相同的. 在实际场景中, 很多时候正例和负例错分的“惩罚”代价是不同的 (参考教材 2.3.4 节). 比如考虑癌症诊断问题, 将一个确实患有癌症的人误分类为健康人, 以及将健康人误分类为患有癌症, 产生的错误影响以及代价不应该认为是等同的. 所以对负例分类错误的样本 (即 false positive) 施加  $k > 0$  倍于正例中被分错的样本的“惩罚”. 对于此类场景下

1. 请给出相应的 SVM 优化问题.
2. 请给出相应的对偶问题, 要求详细的推导步骤, 如 KKT 条件等.

**解:**

1. 我们对每个样例  $(\mathbf{x}_i, y_i)$  附上一个代价系数  $k_i$ ,

$$\text{其中 } k_i = \begin{cases} 1, & y_i = +1 \\ k, & y_i = -1 \end{cases} \text{ 也即 } k_i = 1 - \frac{1}{2}(k-1)(y_i-1).$$

然后相应的 SVM 优化问题即可改为

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^m k_i \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i \in [m] \end{aligned}$$

2. 通过拉格朗日乘子法得到拉格朗日函数

$$L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) = \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^m k_i \xi_i + \sum_{i=1}^m \alpha_i (1 - \xi_i - y_i(\mathbf{w}^T \mathbf{x}_i + b)) - \sum_{i=1}^m \mu_i \xi_i$$

其中  $\alpha_i \geq 0, \mu_i \geq 0$  是拉格朗日乘子.

令  $L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu})$  对  $\mathbf{w}, b, \xi_i$  的偏导等于零可得

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

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

$$k_i C = \alpha_i + \mu_i$$

将上三式逐步带入有

$$\begin{aligned} & L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) \\ &= \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^m k_i \xi_i + \sum_{i=1}^m \alpha_i (1 - \xi_i - y_i(\mathbf{w}^T \mathbf{x}_i + b)) - \sum_{i=1}^m \mu_i \xi_i \\ &= \frac{1}{2} \|\mathbf{w}\|_2^2 + \sum_{i=1}^m \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)) + C \sum_{i=1}^m k_i \xi_i - \sum_{i=1}^m \alpha_i \xi_i - \sum_{i=1}^m \mu_i \xi_i \\ &= -\frac{1}{2} \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i^T \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^m \alpha_i + \sum_{i=1}^m k_i C \xi_i - \sum_{i=1}^m \alpha_i \xi_i - \sum_{i=1}^m \mu_i \xi_i \\ &= -\frac{1}{2} \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i^T \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^m \alpha_i + \sum_{i=1}^m (k_i C - \alpha_i - \mu_i) \xi_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \end{aligned}$$

又因为  $\alpha_i \geq 0, \mu_i \geq 0, k_i C = \alpha_i + \mu_i$ ,



消去  $\mu_i$  即可得到约束条件  $0 \leq \alpha_i \leq k_i C$ .

因此对偶问题为

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

其中  $k_i = \begin{cases} 1, & y_i = +1 \\ k, & y_i = -1 \end{cases}$  也即  $k_i = 1 - \frac{1}{2}(k-1)(y_i-1)$ .

根据 (1) 中的原优化问题所需满足的条件  $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$  可知 KKT 条件  $\alpha_i(y_i f(\mathbf{x}_i) - 1 + \xi_i) = 0$ , 根据  $\xi_i \geq 0$  可知  $\mu_i \xi_i = 0$ .

因此, KKT 条件要求

$$\begin{cases} \alpha_i \geq 0, \mu_i \geq 0, \\ y_i f(\mathbf{x}_i) - 1 + \xi_i \geq 0, \\ \alpha_i(y_i f(\mathbf{x}_i) - 1 + \xi_i) = 0, \\ \xi_i \geq 0, \mu_i \xi_i = 0. \end{cases}$$

#### 四. (20 points) 核函数

教材 6.3 节介绍了 Mercer 定理, 说明对于一个二元函数  $k(\cdot, \cdot)$ , 当且仅当对任意  $m$  和  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 它对应的 Gram 矩阵 (核矩阵) 是半正定的时, 它是一个有效的核函数. 核矩阵  $\mathbf{K}$  中的元素为  $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ . 请根据 Mercer 定理证明以下核函数是有效的.

1.  $\kappa_3 = a_1 \kappa_1 + a_2 \kappa_2$ , 其中  $a_1, a_2 \geq 0$ .
2.  $f(\cdot)$  是任意实值函数, 由  $\kappa_4(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$  定义的  $\kappa_4$ .
3. 由  $\kappa_5(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}') \kappa_2(\mathbf{x}, \mathbf{x}')$  定义的  $\kappa_5$ .
4. 由  $\kappa_6(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})\kappa_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$  定义的  $\kappa_6$

解:

1. 对于任意  $m$  和  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ,

因为  $k_1$  和  $k_2$  是核函数, 因此其对应的核矩阵  $\mathbf{K}_1$  和  $\mathbf{K}_2$  是半正定矩阵,

即有  $\mathbf{y}^T \mathbf{K}_1 \mathbf{y} \geq 0$  与  $\mathbf{y}^T \mathbf{K}_2 \mathbf{y} \geq 0$ , 对于任何  $m$  维向量  $\mathbf{y}$ .

因为  $\kappa_3 = a_1 \kappa_1 + a_2 \kappa_2$ ,

所以有  $K_{ij}^3 = \kappa_3(\mathbf{x}_i, \mathbf{x}_j) = a_1 \kappa_1(\mathbf{x}_i, \mathbf{x}_j) + a_2 \kappa_2(\mathbf{x}_i, \mathbf{x}_j)$

因此有  $\mathbf{K}_3 = a_1 \mathbf{K}_1 + a_2 \mathbf{K}_2$ .

则我们有  $\mathbf{y}^T \mathbf{K}_3 \mathbf{y} = \mathbf{y}^T (a_1 \mathbf{K}_1 + a_2 \mathbf{K}_2) \mathbf{y} = a_1 \mathbf{y}^T \mathbf{K}_1 \mathbf{y} + a_2 \mathbf{y}^T \mathbf{K}_2 \mathbf{y} \geq 0$

所以可知  $\mathbf{K}_3$  也是半正定矩阵,  $\kappa_3$  核函数有效.

2. 对于任意  $m$  和  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 令

$$\mathbf{G} = \begin{pmatrix} f(\mathbf{x}_1) & f(\mathbf{x}_2) & \cdots & f(\mathbf{x}_m) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

则有

$$\begin{aligned} \mathbf{G}^T \mathbf{G} &= \begin{pmatrix} f(\mathbf{x}_1)f(\mathbf{x}_1) & f(\mathbf{x}_1)f(\mathbf{x}_2) & \cdots & f(\mathbf{x}_1)f(\mathbf{x}_m) \\ f(\mathbf{x}_2)f(\mathbf{x}_1) & f(\mathbf{x}_2)f(\mathbf{x}_2) & \cdots & f(\mathbf{x}_2)f(\mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ f(\mathbf{x}_m)f(\mathbf{x}_1) & f(\mathbf{x}_m)f(\mathbf{x}_2) & \cdots & f(\mathbf{x}_m)f(\mathbf{x}_m) \end{pmatrix} \\ &= \begin{pmatrix} \kappa_4(\mathbf{x}_1, \mathbf{x}_1) & \kappa_4(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa_4(\mathbf{x}_1, \mathbf{x}_m) \\ \kappa_4(\mathbf{x}_2, \mathbf{x}_1) & \kappa_4(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa_4(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_4(\mathbf{x}_m, \mathbf{x}_1) & \kappa_4(\mathbf{x}_m, \mathbf{x}_2) & \cdots & \kappa_4(\mathbf{x}_m, \mathbf{x}_m) \end{pmatrix} \\ &= \mathbf{K}_4 \end{aligned}$$

即  $\mathbf{K}_4 = \mathbf{G}^T \mathbf{G}$ .

则有  $\mathbf{y}^T \mathbf{K}_4 \mathbf{y} = \mathbf{y}^T \mathbf{G}^T \mathbf{G} \mathbf{y} = (\mathbf{G} \mathbf{y})^T (\mathbf{G} \mathbf{y}) \geq 0$ , 对于任何  $m$  维向量  $\mathbf{y}$ .

所以可知  $\mathbf{K}_4$  也是半正定矩阵,  $\kappa_4$  核函数有效.

3. 对于任意  $m$  和  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ,

因为  $\kappa_1$  和  $\kappa_2$  是核函数, 我们可知存在  $\phi^{(1)}$  和  $\phi^{(2)}$  使得

$$\kappa_1(\mathbf{x}, \mathbf{x}') = \phi^{(1)}(\mathbf{x})^T \phi^{(1)}(\mathbf{x}') = \sum_i \phi_i^{(1)}(\mathbf{x}) \phi_i^{(1)}(\mathbf{x}')$$

$$\kappa_2(\mathbf{x}, \mathbf{x}') = \phi^{(2)}(\mathbf{x})^T \phi^{(2)}(\mathbf{x}') = \sum_i \phi_i^{(2)}(\mathbf{x}) \phi_i^{(2)}(\mathbf{x}')$$

因此有

$$\begin{aligned} \kappa_5(\mathbf{x}, \mathbf{x}') &= \kappa_1(\mathbf{x}, \mathbf{x}') \kappa_2(\mathbf{x}, \mathbf{x}') \\ &= \sum_i \phi_i^{(1)}(\mathbf{x}) \phi_i^{(1)}(\mathbf{x}') \sum_j \phi_j^{(2)}(\mathbf{x}) \phi_j^{(2)}(\mathbf{x}') \\ &= \sum_i \sum_j (\phi_i^{(1)}(\mathbf{x}) \phi_j^{(2)}(\mathbf{x})) (\phi_i^{(1)}(\mathbf{x}') \phi_j^{(2)}(\mathbf{x}')) \\ &= \sum_{i,j} \phi_{i,j}^{(5)}(\mathbf{x}) \phi_{i,j}^{(5)}(\mathbf{x}') \\ &= \phi^{(5)}(\mathbf{x})^T \phi^{(5)}(\mathbf{x}') \end{aligned}$$

因此对于任意  $m$  维向量  $\mathbf{y}$ , 有

$$\begin{aligned} \mathbf{y}^T \mathbf{K}_5 \mathbf{y} &= \sum_{i=1}^m \sum_{j=1}^m y_i \kappa_5(\mathbf{x}_i, \mathbf{x}_j) y_j \\ &= \sum_{i=1}^m \sum_{j=1}^m y_i \phi^{(5)}(\mathbf{x}_i)^T \phi^{(5)}(\mathbf{x}_j) y_j \\ &= \sum_{i=1}^m (y_i \phi^{(5)}(\mathbf{x}_i))^T \sum_{j=1}^m (y_j \phi^{(5)}(\mathbf{x}_j)) \\ &= \left( \sum_{i=1}^m y_i \phi^{(5)}(\mathbf{x}_i) \right)^T \left( \sum_{i=1}^m y_i \phi^{(5)}(\mathbf{x}_i) \right) \geq 0 \end{aligned}$$

因此  $\mathbf{K}_5$  也是半正定矩阵,  $\kappa_5$  核函数有效.

4. 对于任意  $m$  和  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , 令

$$\mathbf{G} = \begin{pmatrix} f(\mathbf{x}_1) & 0 & \cdots & 0 \\ 0 & f(\mathbf{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\mathbf{x}_m) \end{pmatrix}$$

则有

$$\begin{aligned} & \mathbf{G}^\top \mathbf{K}_1 \mathbf{G} \\ &= \begin{pmatrix} f(\mathbf{x}_1)\kappa_1(\mathbf{x}_1, \mathbf{x}_1)f(\mathbf{x}_1) & \cdots & f(\mathbf{x}_1)\kappa_1(\mathbf{x}_1, \mathbf{x}_m)f(\mathbf{x}_m) \\ f(\mathbf{x}_2)\kappa_1(\mathbf{x}_2, \mathbf{x}_1)f(\mathbf{x}_1) & \cdots & f(\mathbf{x}_2)\kappa_1(\mathbf{x}_2, \mathbf{x}_m)f(\mathbf{x}_m) \\ \vdots & \ddots & \vdots \\ f(\mathbf{x}_m)\kappa_1(\mathbf{x}_m, \mathbf{x}_1)f(\mathbf{x}_1) & \cdots & f(\mathbf{x}_m)\kappa_1(\mathbf{x}_m, \mathbf{x}_m)f(\mathbf{x}_m) \end{pmatrix} \\ &= \begin{pmatrix} \kappa_6(\mathbf{x}_1, \mathbf{x}_1) & \kappa_6(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa_6(\mathbf{x}_1, \mathbf{x}_m) \\ \kappa_6(\mathbf{x}_2, \mathbf{x}_1) & \kappa_6(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa_6(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_6(\mathbf{x}_m, \mathbf{x}_1) & \kappa_6(\mathbf{x}_m, \mathbf{x}_2) & \cdots & \kappa_6(\mathbf{x}_m, \mathbf{x}_m) \end{pmatrix} \\ &= \mathbf{K}_6 \end{aligned}$$

则有  $\mathbf{y}^\top \mathbf{K}_6 \mathbf{y} = \mathbf{y}^\top \mathbf{G}^\top \mathbf{K}_1 \mathbf{G} \mathbf{y} = (\mathbf{G} \mathbf{y})^\top \mathbf{K}_1 (\mathbf{G} \mathbf{y}) \geq 0$ , 对于任何  $m$  维向量  $\mathbf{y}$ .

所以可知  $\mathbf{K}_6$  也是半正定矩阵,  $\kappa_6$  核函数有效.

### 五. (20 points) 主成分分析

$\mathbf{x} \in \mathbb{R}^d$  是一个随机向量, 其均值和协方差分别是  $\boldsymbol{\mu} = \mathbb{E}(\mathbf{x}) \in \mathbb{R}^d$ ,  $\boldsymbol{\Sigma} = \mathbb{E}(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top \in \mathbb{R}^{d \times d}$ . 定义随机变量  $\{y_i = \mathbf{u}_i^\top \mathbf{x} + a_i \in \mathbb{R}, i = 1, \dots, d' \leq d\}$  为  $\mathbf{x}$  的主成分, 其中  $\mathbf{u}_i \in \mathbb{R}^d$  是单位向量 ( $\mathbf{u}_i^\top \mathbf{u}_i = 1$ ),  $a_i \in \mathbb{R}$ ,  $\{y_i\}_{i=1}^{d'}$  是互不相关的零均值随机变量, 它们的方差满足  $\text{var}(y_1) \geq \text{var}(y_2) \geq \cdots \geq \text{var}(y_{d'})$ . 假设  $\boldsymbol{\Sigma}$  没有重复的特征值.

1. 请证明  $\{a_i = -\mathbf{u}_i^\top \boldsymbol{\mu}\}_{i=1}^{d'}$ .

2. 请证明  $\mathbf{u}_1$  是  $\Sigma$  最大的特征值对应的特征向量. (提示: 写出要最大化的目标函数, 写出约束条件, 使用拉格朗日乘子法)
3. 请证明  $\mathbf{u}_2^\top \mathbf{u}_1 = 0$ , 且  $\mathbf{u}_2$  是  $\Sigma$  第二大特征值对应的特征向量. (提示: 由  $\{y_i\}_{i=1}^d$  是互不相关的零均值随机变量可推出  $\mathbf{u}_2^\top \mathbf{u}_1 = 0$ , 可作为第二小问的约束条件之一)
4. 通过 PCA 进行降维, 得到的随机变量满足  $\text{var}(y_1) \geq \text{var}(y_2) \geq \dots \geq \text{var}(y_d)$ , 也就是降维后的数据在不同维度上有不同的方差, 从而导致不同维度的数值范围差异很大, 如果想要降维后的样本在不同维度具有大致相同的数值范围, 应该怎么做?

解:

1. 因为  $\{y_i = \mathbf{u}_i^\top \mathbf{x} + a_i \in \mathbb{R}\}_{i=1}^{d'}$  是互不相关的零均值随机变量, 因此  $\mathbb{E}[y_i] = \mathbb{E}[\mathbf{u}_i^\top \mathbf{x} + a_i] = \mathbf{u}_i^\top \mathbb{E}[\mathbf{x}] + a_i = 0$ .  
又因为  $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$ ,  
所以有  $a_i = -\mathbf{u}_i^\top \boldsymbol{\mu}$ .

2. 对于随机变量  $\{y_i\}_{i=1}^{d'}$  来说, 方差为

$$\begin{aligned} \text{var}(y_i) &= \mathbb{E}[y_i^2] - (\mathbb{E}[y_i])^2 \\ &= \mathbb{E}[y_i^2] - 0 \\ &= \mathbb{E}[\mathbf{u}_i^\top (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{u}_i] \\ &= \mathbf{u}_i^\top \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top] \mathbf{u}_i \\ &= \mathbf{u}_i^\top \Sigma \mathbf{u}_i \end{aligned}$$

我们要最大化的目标函数即为  $\sum_{i=1}^{d'} \text{var}(y_i) = \sum_{i=1}^{d'} \mathbf{u}_i^\top \Sigma \mathbf{u}_i$ . 具体优化问题为

$$\begin{aligned} \min_{\mathbf{u}_i} \quad & - \sum_{i=1}^{d'} \mathbf{u}_i^\top \Sigma \mathbf{u}_i \\ \text{s.t.} \quad & \mathbf{u}_i^\top \mathbf{u}_i = 1, i = 1, \dots, d' \end{aligned}$$

注：在 (3) 我们还会证明还有约束条件  $\mathbf{u}_i^T \mathbf{u}_j = 0$ , 其中  $i \neq j$ . 但是这里我们先不加上这个约束条件.

构造拉格朗日函数得

$$L(\mathbf{u}_i, \lambda_i) = - \sum_{i=1}^{d'} \mathbf{u}_i^T \Sigma \mathbf{u}_i + \sum_{i=1}^{d'} \lambda_i (\mathbf{u}_i^T \mathbf{u}_i - 1)$$

对  $\mathbf{u}_i$  求偏导并令其等于零即有

$$\frac{\partial L(\mathbf{u}_i, \lambda_i)}{\partial \mathbf{u}_i} = -2\Sigma \mathbf{u}_i + 2\lambda_i \mathbf{u}_i = 0$$

即有

$$\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

因此  $\lambda_i$  是  $\Sigma$  的特征值,  $\mathbf{u}_i$  是  $\Sigma$  和  $\lambda_i$  的对应的特征向量.

我们将  $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$  带入  $\text{var}(y_i) = \mathbf{u}_i^T \Sigma \mathbf{u}_i$  则有

$$\text{var}(y_i) = \mathbf{u}_i^T \Sigma \mathbf{u}_i = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i \mathbf{u}_i^T \mathbf{u}_i = \lambda_i$$

即  $y_i$  的方差  $\text{var}(y_i)$  大小等于其对应的特征值.

又因为  $\text{var}(y_1) \geq \text{var}(y_2) \geq \dots \geq \text{var}(y_{d'})$ , 即有  $y_1$  的方差最大, 因此  $y_1$  中的  $\mu_1$  是  $\Sigma$  最大的特征值  $\lambda_1$  对应的特征向量.

3. 因为  $\{y_i = \mathbf{u}_i^T \mathbf{x} + a_i \in \mathbb{R}\}_{i=1}^{d'}$  是互不相关的零均值随机变量, 因此  $\mathbb{E}[y_i y_j] = \mathbb{E}[y_i] \mathbb{E}[y_j] = 0$ , 其中  $i \neq j$ , 并且我们有 (1) 中的  $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$ , 即有

$$\begin{aligned} \mathbb{E}[y_i y_j] &= \mathbb{E}[(\mathbf{u}_i^T \mathbf{x} + a_i)(\mathbf{u}_j^T \mathbf{x} + a_j)] \\ &= \mathbf{u}_i^T \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] \mathbf{u}_j \\ &= \mathbf{u}_i^T \Sigma \mathbf{u}_j \\ &= \lambda_i \mathbf{u}_i^T \mathbf{u}_j \\ &= \lambda_j \mathbf{u}_i^T \mathbf{u}_j \\ &= \mathbb{E}[y_i] \mathbb{E}[y_j] = 0 \end{aligned}$$

因为  $\Sigma$  没有重复的特征值, 因此  $\lambda_i \neq \lambda_j$  即  $\lambda_i - \lambda_j \neq 0$ , 所以

$$\lambda_i \mathbf{u}_i^T \mathbf{u}_j - \lambda_j \mathbf{u}_i^T \mathbf{u}_j = (\lambda_i - \lambda_j) \mathbf{u}_i^T \mathbf{u}_j = 0 - 0 = 0$$

因此有

$$\mathbf{u}_i^T \mathbf{u}_j = 0$$

因此  $\mathbf{u}_i$  和  $\mathbf{u}_j$  是相互正交的特征单位向量, 他们对应的特征值  $\lambda_i$  和  $\lambda_j$  也互不相同.

又因为  $\text{var}(y_1) \geq \text{var}(y_2) \geq \cdots \geq \text{var}(y_{d'})$ ,

因此有  $\lambda_1 > \lambda_2 > \cdots > \lambda_{d'}$ ,

即有答案所需的  $\mathbf{u}_2^T \mathbf{u}_1 = 0$  且  $\mathbf{u}_2$  是  $\Sigma$  第二大特征值对应的特征向量.

4. 我们可以对降维后的随机变量  $\{y_i\}_{i=1}^{d'}$  都除去对应的标准差, 即方差的开方, 进行”归一化”.

$$\text{即令 } y'_i = \frac{y_i}{\sqrt{\text{var}(y_i)}} = \frac{y_i}{\sqrt{\lambda_i}}.$$

则我们求  $\{y'_i\}_{i=1}^{d'}$  的方差即有  $\text{var}(y'_i) = \left(\frac{1}{\sqrt{\text{var}(y_i)}}\right)^2 \cdot \text{var}(y_i) = 1$ .

这样, 降维后的  $\{y'_i\}_{i=1}^{d'}$  方差均为 1, 也就有大致相同的数值范围了.