ML-04

201300086史浩男

第二题附代码Vectorization.py

一、神经网络基础

给定训练集 $D=\{(\boldsymbol{x}_1,\boldsymbol{y}_1),(\boldsymbol{x}_2,\boldsymbol{y}_2),...,(\boldsymbol{x}_m,\boldsymbol{y}_m)\}$. 其中 $\boldsymbol{x}_i\in\mathbb{R}^d$, $\boldsymbol{y}_i\in\mathbb{R}^l$ 表示输入示例由 d 个属性描述, 输出 l 维实值向量. 图 1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 θ_j 表示, 隐层第 h 个神经元的阈值用 γ_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.

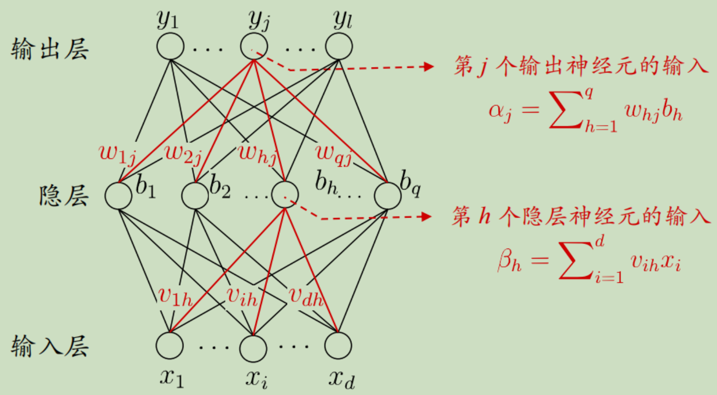

Figure 1: 多层神经网络（教材图 5.7）

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 $(\boldsymbol{x},y)$ 有

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

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

(1)Sigmoid

已知 $\hat{y}_j=f\left(\beta_j-\theta_j\right)$, 且 $f'(x)=f(x)(1-f(x))$, 所以

$$\begin{aligned}\frac{\partial\ell(y,\hat{y}_1)}{\partial\beta_1}&=\frac{\partial\ell(y,\hat{y}_1)\partial\hat{y}_1}{\partial\hat{y}_1\partial\beta_1}\\&=\left(\frac{1-y}{1-\hat{y}_1}-\frac{y}{\hat{y}_1}\right)f'\left(\beta_1-\theta_1\right)\\&=\left(\frac{1-y}{1-\hat{y}_1}-\frac{y}{\hat{y}_1}\right)\hat{y}_1\left(1-\hat{y}_1\right)\\&=\hat{y}_1-y_1\end{aligned}$$

2. 当 $l > 1$, 网络的预测结果为 $\hat{\boldsymbol{y}} \in \mathbb{R}^l$, 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 $\boldsymbol{y} \in \{0,1\}^l$, 有 $y_i = 1$, $y_j = 0, j \neq i$. 对于一个训练样本 $(\boldsymbol{x}, \boldsymbol{y})$, 交叉熵损失函数 $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$ 的定义如下

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

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

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

易见 Softmax 函数输出的 $\hat{\boldsymbol{y}}$ 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 $\frac{\partial \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \beta_j}$,

(2)Softmax多分类

不能只对维度j求偏导，因为每个$\hat{y}_k$中都有β_j

对于$\forall k \neq j$:

$$\begin{aligned}
\frac{\partial \hat{y}_k}{\partial \beta_j} &= \frac{\partial\left(\frac{e^{\beta_k}}{\sum_{i=1}^{l} e^{\beta_i}}\right)}{\partial \beta_j} \\
&= -\hat{y}_k \frac{e^{\beta_j}}{\sum_{i=1}^{l} e^{\beta_i}} \\
&= -\hat{y}_k e^{\beta_j} \frac{\hat{y}_j}{e^{\beta_j}} \\
&= -\hat{y}_k \hat{y}_j
\end{aligned}$$

因此求偏导:

$$\begin{aligned}
\frac{\partial \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \beta_j} &= \frac{\partial \ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) \partial \hat{y}_j}{\partial \hat{y}_j \partial \beta_j} + \sum_{k \neq j} \frac{\partial \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial \beta_j} \\
&= -\frac{y_j}{\hat{y}_j} \times \hat{y}_j(1 - \hat{y}_j) + \sum_{k \neq j} \frac{y_k}{\hat{y}_k} \times \hat{y}_k \hat{y}_j \\
&= -y_j + y_j \hat{y}_j + \sum_{k \neq j} y_k \hat{y}_j \\
&= \sum_{k=1}^{l} y_k \hat{y}_j - y_j \\
&= \hat{y}_j - y_j
\end{aligned}$$

(3)二分类中使用`Softmax`和`Sigmoid`的联系与区别

联系

把$j = 1$带入第二问的结果中，可以发现`Softmax`和`Sigmoid`的求偏导结果相同

因此二者用于二分类问题时的结果是大致相同的，并且两个类别概率的和为 1

区别

`Softmax`计算的是一个比重，是一个针对输出结果归一化的过程，对两个类别均输出对应的概率。

`Sigmoid` 只是对每一个输出值进行非线性化，只是一个非线性激活过程，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。

所以`softmax`一般用于多分类的结果，大多数用于网络的最后一层。而`sigmoid`是原本一种隐层之间的激活函数，效果比其他激活函数差，一般只会出现在二分类的输出层中，与0 1真实标签配合使用。

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

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

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.

(4)KL散度与交叉熵损失

交叉熵损失函数：

$$\ell(P,Q)=-\sum_{x\in\mathcal{X}}P\left(x\right)\log\left(Q\left(x\right)\right)$$

信息熵函数：

$$Ent(P)=\sum_{x\in\mathcal{X}}P\left(x\right)\log\left(P\left(x\right)\right)$$

所以KL散度可以表示为

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

分析

如果$P(x)$代表了输入样本中的$\boldsymbol{y}$，则样本输入不变时，$P(x)$与$Ent(P)$不变，都可以视作常数

因此 $D_{\mathrm{KL}}(P\|Q)$ 与 $\ell(\boldsymbol{y},\hat{\boldsymbol{y}})$ 只是在常数上有区别，最小化 $D_{\mathrm{KL}}(P\|Q)$ 等价于最小化$\ell(\boldsymbol{y},\hat{\boldsymbol{y}})$.

二、运算向量化

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 $\boldsymbol{X}\in\mathbb{R}^{m\times d}$，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 $\boldsymbol{D}\in\mathbb{R}^{m\times m}$，两个向量之间的欧式距离定义为 $\|\boldsymbol{x}-\boldsymbol{y}\|_2=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2}$. 求距离矩阵可以通过循环的方式，即plain_distance_function中实现的方法；

```python
import numpy as np

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



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

(1)

将 $\sum_{i=1}^{d}(x_i-y_i)^2$ 拆分成 $\sum_{i=1}^{d}x_i^2+\sum_{i=1}^{d}y_i^2-2\sum_{i=1}^{d}x_iy_i$，然后分别计算三个矩阵，再相加。

X是随机$m*d$维矩阵，m和d指定

代码如下：

```python
def matrix_distance_function(X: np.ndarray):
    # (xi - xj)^2 = xi^2 + xj^2 - 2 xi * xj
    ones = np.ones(X.shape).T
    M_i = np.square(X) @ ones#从m*d得到m*m的行平方和，每行都相同
    M_j = M_i.T
    M_ij = X @ X.T
    M = M_i + M_j - 2 * M_ij + 1e-10# 加1e-10防止对负数开根号
    return np.sqrt(M)
```

结果与分析：

- 分别针对（10，10）和（1000，1000）的矩阵规模尝试`plain`方法和`matrix`方法，
- 其中小规模矩阵计算的是运行100次总时间
- 具体时间计算时没有计算生成随机矩阵X所需时间

程序运行时间如下

```
100 times for (10,10) scale using plain function: 0.14656734466552734
100 times for (10,10) scale using matrix function: 0.014003038406372O7
1 times for (1000,1000) scale using plain function: 29.628735065460205
1 times for (1000,1000) scale using matrix function: 0.17103886604309082
```

- 小规模矩阵（10，10）时，`matrix`方法所需时间大概是`plain`的$\frac{1}{10}$
- 大规模矩阵（1000，1000）时，`matrix`方法所需时间大概是`plain`的$\frac{1}{150}$

我们发现`matrix`方法对于`plain`在不同规模下都有绝对优势。

这是因为`matrix`方法中分离了x和y内部的平方计算，计算出每个示例内部的平方和后复制了d次，减少了大量重复计算，因此我们有理由相信任意矩阵规模`matrix`方法都会有性能的提升

2. 输入一个矩阵 $\boldsymbol{X}\in\mathbb{R}^{m\times d}$，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 $\boldsymbol{p}=\{p_1,p_2,...,p_m\}$ 进行重新排列. 即输出一个新的矩阵 $\boldsymbol{X}'$，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 $\boldsymbol{X}'=\mathrm{perm}(\boldsymbol{X})$，已知梯度 $\frac{\partial\ell}{\partial\boldsymbol{X}'}$，需要计算 $\frac{\partial\ell}{\partial\boldsymbol{X}}$. 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.

```python
import numpy as np

def plain_permutation_function(X, p):
    # 初始化结果矩阵，其中每一行对应一个样本
    permuted_X = np.zeros_like(X)
    for i in range(X.shape[0]):
        # 采用循环的方式对每一个样本进行重排列
        permuted_X[i] = X[p[i]]
    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）

(2)

根据任意排列p生成变换矩阵P的方法：

```
def matrix_permutation_function(X: np.ndarray, p: np.ndarray):
    M_per = np.zeros((X.shape[0], X.shape[0]))
    for i in range(X.shape[0]):
        M_per[i, p[i]] = 1
    return M_per @ X
```

函数返回的PX即为我们需要的结果

结果与分析：

- 分别针对（10，10）和（2000，2000）的矩阵规模尝试`plain`方法和`matrix`方法，
- 其中小规模矩阵计算的是运行100次总时间，大规模矩阵计算的是运行10次总时间
- 具体时间计算时没有计算生成随机矩阵X和随机序列p所需时间

程序运行时间如下

```
100 times for (10,10) scale using plain function: 0.007001638412475586
100 times for (10,10) scale using matrix function: 0.0010001659393310547
10 times for (2000,2000) scale using plain function: 0.2680797576904297
10 times for (2000,2000) scale using matrix function: 8.44570517539978
```

- 小规模矩阵（10，10）时，`matrix`方法所需时间大概是`plain`的$\frac{1}{7}$
- 大规模矩阵（1000，1000）时，`matrix`方法所需时间大概是`plain`的30倍

我们发现`matrix`方法对于`plain`在小规模下有一些优势，但在输入规模增加后开始显露出越来越大的劣势

这是因为相比于`plain`方法的朴素直接计算，`matrix`方法需要先生成$m \times m$的巨大变换矩阵P，然后再将两个巨大的矩阵相乘来得到结果，使得计算开销随着矩阵规模的扩大越发不可控

三、支持向量机

考虑标准的 SVM 优化问题如下 (即教材公式 (6.35)),

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

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.

负例分类错误：$y_i = -1, \ w^Tx + b > 1$

正例分类错误：$y_i = 1, \ w^Tx + b < -1$

0/1损失函数：

$$\ell_{0/1}(y,z) = \begin{cases} k, & \text{if } z < 0, y = -1 \\ 1, & \text{if } z < 0, y = 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\text{简化后：}\ell_{0/1}(y,z)=\begin{cases}\frac{k+1-y(k-1)}{2}, & \text{if } z<0\\ 0, & \text{otherwise}\end{cases}$$

(1)SVM优化问题

$$\begin{array}{ll}\min_{\boldsymbol{w},b,\xi_i} & \frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}\frac{k+1-y_i(k-1)}{2}\xi_i\\ \text{s.t.} & y_i\left(\boldsymbol{w}^\top\boldsymbol{x}_i+b\right)\geq 1-\xi_i\\ & \xi_i\geq 0, i\in[m]\end{array}$$

(2)对偶问题

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

对$\boldsymbol{w},b,\xi_i$ 求偏导等于零可得

$$\begin{array}{l}w=\sum_{i=1}^{m}\alpha_iy_i\boldsymbol{x}_i\\ \sum_{i=1}^{m}\alpha_iy_i=0\\ \frac{k+1-y_i(k-1)}{2}C=\alpha_i+\mu_i\end{array}$$

代入化简

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

约束条件消去除α_i,y_i外其他项

$$\sum_{i=1}^{m}\alpha_iy_i=0$$

$$\frac{k+1-y_i(k-1)}{2}C-\alpha_i\geq 0$$

得到对偶问题

$$\begin{aligned}\max_{\boldsymbol{\alpha}}&\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_jy_iy_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j\\ \text{s.t.}&\sum_{i=1}^{m}\alpha_iy_i=0\\ &0\leq\alpha_i\leq\frac{k+1-y_i(k-1)}{2}C\end{aligned}$$

对任意训练样本 $(\boldsymbol{x}_i,y_i)$总有 $\alpha_i=0$ 或 $y_if(\boldsymbol{x}_i)=1-\xi_i$

- 若 $\alpha_i=0$, 该样本不会对 $f(\boldsymbol{x})$有任何影响;
- 若 $y_if(\boldsymbol{x}_i)=1-\xi_i$, 该样本是支持向量。
  - 若 $\alpha_i<\frac{k+1-y_i(k-1)}{2}C$ 则 $\mu_i>0$，$\xi_i=0$, 该样本恰在最大间隔边界上;
  - 若 $\alpha_i=\frac{k+1-y_i(k-1)}{2}C$ 则 $\mu_i=0$, 若 $\xi_i\leqslant 1$ 则该样本落在最大间隔内部, 若 $\xi_i>1$ 则该样本被错误分类

KKT条件

$$\begin{cases}\alpha_i\geq 0,\mu_i\geq 0\\ y_if(\boldsymbol{x}_i)-1+\xi_i\geq 0\\ \alpha_i\left(y_if(\boldsymbol{x}_i)-1+\xi_i\right)=0\\ \xi_i\geq 0,\mu_i\xi_i=0\end{cases}$$

四、核函数拓展性

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 $\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$，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 $\boldsymbol{K}$ 中的元素为 $K_{ij}=\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$．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 $\kappa_4(\boldsymbol{x},\boldsymbol{x}')=f(\boldsymbol{x})f(\boldsymbol{x}')$ 定义的 κ_4．
3. 由 $\kappa_5(\boldsymbol{x},\boldsymbol{x}')=\kappa_1(\boldsymbol{x},\boldsymbol{x}')\kappa_2(\boldsymbol{x},\boldsymbol{x}')$ 定义的 κ_5．
4. 由 $\kappa_6(\boldsymbol{x},\boldsymbol{x}')=f(\boldsymbol{x})\kappa_1(\boldsymbol{x},\boldsymbol{x}')f(\boldsymbol{x}')$ 定义的 κ_6

设κ_i对应的核矩阵为K_i

(1)

由于κ_1,κ_2是核函数，其核矩阵半正定，即对任意实非零m维向量x有：

$$x^TK_1x\geq 0,x^TK_2x\geq 0$$

由于$\kappa_3=a_1\kappa_1+a_2\kappa_2$，则对任意$a_1,a_2\geq 0$，满足：

$$a_1x^TK_1x+a_2x^TK_2x=x^TK_3x\geq 0$$

所以K_3半正定，κ_3有效

(2)

$$\begin{aligned}K_4&=\begin{pmatrix}\kappa_4(\boldsymbol{x}_1,\boldsymbol{x}_1)&\kappa_4(\boldsymbol{x}_1,\boldsymbol{x}_2)&\cdots&\kappa_4(\boldsymbol{x}_1,\boldsymbol{x}_m)\\\kappa_4(\boldsymbol{x}_2,\boldsymbol{x}_1)&\kappa_4(\boldsymbol{x}_2,\boldsymbol{x}_2)&\cdots&\kappa_4(\boldsymbol{x}_2,\boldsymbol{x}_m)\\\vdots&\vdots&\ddots&\vdots\\\kappa_4(\boldsymbol{x}_m,\boldsymbol{x}_1)&\kappa_4(\boldsymbol{x}_m,\boldsymbol{x}_2)&\cdots&\kappa_4(\boldsymbol{x}_m,\boldsymbol{x}_m)\end{pmatrix}\\&=(f(\boldsymbol{x}_1),f(\boldsymbol{x}_2),\cdots,f(\boldsymbol{x}_m))^T(f(\boldsymbol{x}_1),f(\boldsymbol{x}_2),\cdots,f(\boldsymbol{x}_m))\end{aligned}$$

因此对任意实非零m维向量x：

$$x^TK_4x=x^T(f(\boldsymbol{x}_1),f(\boldsymbol{x}_2),\cdots,f(\boldsymbol{x}_m))^T(f(\boldsymbol{x}_1),f(\boldsymbol{x}_2),\cdots,f(\boldsymbol{x}_m))x\geq 0$$

所以K_4半正定，κ_4有效

(3)

对任意实非零m维向量y：

$$\begin{aligned}y^TK_5y&=\sum_{i=1}^m\sum_{j=1}^m\kappa_1\left(x_i,x_i^{'}\right)\kappa_2\left(x_j,x_j^{'}\right)y_iy_j\\&=tr(\begin{bmatrix}y_1\kappa_1(x_1,x_1')&y_1\kappa_1(x_1,x_2')&\cdots&y_1\kappa_1(x_1,x_m')\\y_2\kappa_1\left(x_2,x_1^{'}\right)&y_2\kappa_1(x_2,x_2')&\cdots&y_2\kappa_1(x_2,x_m')\\\vdots&\vdots&\ddots&\vdots\\y_m\kappa_1\left(x_m,x_1^{'}\right)&y_n\kappa_1\left(x_m,x_2^{'}\right)&\cdots&y_n\kappa_1(x_m,x_m')\end{bmatrix}\begin{bmatrix}y_1\kappa_2(x_1,x_1')&y_1\kappa_2(x_1,x_2')&\cdots&y_1\kappa_2(x_1,x_m')\\y_2\kappa_2\left(x_2,x_1^{'}\right)&y_2\kappa_2(x_2,x_2')&\cdots&y_2\kappa_2(x_2,x_m')\\\vdots&\vdots&\ddots&\vdots\\y_m\kappa_2\left(x_m,x_1^{'}\right)&y_n\kappa_2\left(x_m,x_2^{'}\right)&\cdots&y_n\kappa_1(x_m,x_m')\end{bmatrix})\\&=tr\left(\begin{bmatrix}y_1&&&\\&y_2&&\\&&\ddots&\\&&&y_m\end{bmatrix}K_1\begin{bmatrix}y_1&&&\\&y_2&&\\&&\ddots&\\&&&y_m\end{bmatrix}K_2^T\right)\end{aligned}$$

由于K_1,K_2是半正定矩阵，所以满足$K_1=C^TC,\quad K_2=D^TD$

$$
\begin{aligned}
y^T K_5 y &= tr\left(\begin{bmatrix} y_1 & & & \\ & y_2 & & \\ & & \ddots & \\ & & & y_m \end{bmatrix} C^T C \begin{bmatrix} y_1 & & & \\ & y_2 & & \\ & & \ddots & \\ & & & y_m \end{bmatrix} D^T D\right) \\
&= tr\left(\left(C\begin{bmatrix} y_1 & & & \\ & y_2 & & \\ & & \ddots & \\ & & & y_m \end{bmatrix} D^T\right)^T \left(C\begin{bmatrix} y_1 & & & \\ & y_2 & & \\ & & \ddots & \\ & & & y_m \end{bmatrix} D^T\right)\right) \\
&\geq 0
\end{aligned}
$$

所以K_5半正定，κ_5有效

(4)

$$
\begin{aligned}
K_6 &= \begin{pmatrix} f(\boldsymbol{x}_1)\kappa_1(\boldsymbol{x}_1,\boldsymbol{x}_1)f(\boldsymbol{x}_1) & \cdots & f(\boldsymbol{x}_1)\kappa_1(\boldsymbol{x}_1,\boldsymbol{x}_m)f(\boldsymbol{x}_m) \\ f(\boldsymbol{x}_2)\kappa_1(\boldsymbol{x}_2,\boldsymbol{x}_1)f(\boldsymbol{x}_1) & \cdots & f(\boldsymbol{x}_2)\kappa_1(\boldsymbol{x}_2,\boldsymbol{x}_m)f(\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ f(\boldsymbol{x}_m)\kappa_1(\boldsymbol{x}_m,\boldsymbol{x}_1)f(\boldsymbol{x}_1) & \cdots & f(\boldsymbol{x}_m)\kappa_1(\boldsymbol{x}_m,\boldsymbol{x}_m)f(\boldsymbol{x}_m) \end{pmatrix} \\
&= \begin{pmatrix} f(\boldsymbol{x}_1) & 0 & \cdots & 0 \\ 0 & f(\boldsymbol{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\boldsymbol{x}_m) \end{pmatrix}^T \begin{pmatrix} \kappa_1(\boldsymbol{x}_1,\boldsymbol{x}_1) & \kappa_1(\boldsymbol{x}_1,\boldsymbol{x}_2) & \cdots & \kappa_1(\boldsymbol{x}_1,\boldsymbol{x}_m) \\ \kappa_1(\boldsymbol{x}_2,\boldsymbol{x}_1) & \kappa_1(\boldsymbol{x}_2,\boldsymbol{x}_2) & \cdots & \kappa_1(\boldsymbol{x}_2,\boldsymbol{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_1(\boldsymbol{x}_m,\boldsymbol{x}_1) & \kappa_1(\boldsymbol{x}_m,\boldsymbol{x}_2) & \cdots & \kappa_1(\boldsymbol{x}_m,\boldsymbol{x}_m) \end{pmatrix} \begin{pmatrix} f(\boldsymbol{x}_1) & 0 & \cdots & 0 \\ 0 & f(\boldsymbol{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\boldsymbol{x}_m) \end{pmatrix}
\end{aligned}
$$

所以K_6可被表示为H^TK_1H

对任意实非零m维向量y:

$$\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}_6\boldsymbol{y} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{K}_1\boldsymbol{H}\boldsymbol{y} = (\boldsymbol{H}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{K}_1(\boldsymbol{H}\boldsymbol{y}) \geq 0$$

五、PCA

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

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

(1)

由于$\{y_i\}_{i=1}^{d'}$是互不相关零均值随机变量，则

$$
\begin{gathered}
\sum_{i=1}^{d'} y_i = \sum_{i=1}^{d'} \boldsymbol{u}_{\boldsymbol{i}}^T\boldsymbol{x} + \sum_{i=1}^{d'} a_i = 0 \\
\mathbb{E}(y_i) = \boldsymbol{u}_{\boldsymbol{i}}^T\mathbb{E}(\boldsymbol{x}) + a_i = \boldsymbol{u}_{\boldsymbol{i}}^T\boldsymbol{\mu} + a_i = 0 \\
a_i = -\boldsymbol{u}_i^\top\boldsymbol{\mu}
\end{gathered}
$$

2. 请证明 $\boldsymbol{u}_1$ 是 $\boldsymbol{\Sigma}$ 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 $\boldsymbol{u}_2^\top\boldsymbol{u}_1 = 0$，且 $\boldsymbol{u}_2$ 是 $\boldsymbol{\Sigma}$ 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 $\boldsymbol{u}_2^\top\boldsymbol{u}_1 = 0$, 可作为第二小问的约束条件之一)

(2)最大特征值即最大方差

我们已知最大方差是$var(y_1)$，需要证明最大特征值即是最大方差

$$\begin{aligned} var(y_i) &= \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2 \\ &= \mathbb{E}((\boldsymbol{u_i}^T(\boldsymbol{x} - \boldsymbol{\mu}))^2) \\ &= \boldsymbol{u_i}^T \mathbb{E}(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{u_i} \\ &= \boldsymbol{u_i}^T \boldsymbol{\Sigma} \boldsymbol{u_i} \end{aligned}$$

因此最大化目标函数为$\sum_{i=1}^{d'} var(y_i) = \sum_{i=1}^{d'} \boldsymbol{u_i}^T \boldsymbol{\Sigma} \boldsymbol{u_i}$，对应最优化问题和约束条件：

$$min_{\boldsymbol{u_i}} \; -\sum_{i=1}^{d'} \boldsymbol{u_i}^T \boldsymbol{\Sigma} \boldsymbol{u_i}$$

$$s.t. \, \boldsymbol{u_i}^T \boldsymbol{u_i} = 1, i \in [d']$$

使用拉格朗日乘子法：

$$L(\boldsymbol{u_i}, \boldsymbol{\lambda}) = -\sum_{i=1}^{d'} \boldsymbol{u_i}^T \boldsymbol{\Sigma} \boldsymbol{u_i} + \boldsymbol{\lambda}(\boldsymbol{u_i}^T \boldsymbol{u_i} - 1)$$

对$\boldsymbol{u_i}$求偏导，令其为0：

$$\boldsymbol{\lambda u_i} - \sum_{i=1}^{d'} \boldsymbol{\Sigma u_i} = 0$$

$$\text{即} \lambda_i \boldsymbol{u_i} = \boldsymbol{\Sigma u_i}$$

所以我们得出λ_i是$\boldsymbol{\Sigma}$的特征值，$\boldsymbol{u_i}$是λ_i对应的特征向量，因此

$$\begin{aligned} var(y_i) &= \boldsymbol{u_i}^T \boldsymbol{\Sigma} \boldsymbol{u_i} \\ &= \lambda_i \boldsymbol{u_i}^T \boldsymbol{u_i} \\ &= \lambda_i \end{aligned}$$

再由于$\operatorname{var}(y_1) \geq \operatorname{var}(y_2) \geq \cdots \geq \operatorname{var}(y_{d'})$，因此$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{d'}$

因此最大特征值λ_1对应的向量是$\boldsymbol{u_1}$

(3)特征向量正交

由于$\{y_i\}_{i=1}^{d'}$是互不相关零均值随机变量，则

$$\mathbb{E}(y_i y_j) = \mathbb{E}(y_i)\mathbb{E}(y_j) = 0$$

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

由于$\boldsymbol{\Sigma}$没有重复特征根，所以对$\forall i, j$，$\lambda_i \neq \lambda_j$，所以只能$\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{u}_j = 0$

在(2)中已经证明：第k大的特征根是λ_k，对应的特征向量是$\boldsymbol{u}_k$

4. 通过 PCA 进行降维，得到的随机变量满足 $\operatorname{var}(y_1) \geq \operatorname{var}(y_2) \geq \cdots \geq \operatorname{var}(y_d)$，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？

(4)标准化

只需要对降维后的随机变量进行标准化操作：

$$y_i' = \frac{y_i}{\sqrt{\operatorname{var}(y_i)}} = \frac{y_i}{\sqrt{\lambda_i}}$$

结果可以保证降维后方差都为1：

$$\operatorname{var}(y_i') = \left(\frac{1}{\sqrt{\operatorname{var}(y_i)}}\right)^2 \cdot \operatorname{var}(y_i) = 1$$