模式识别实验报告

专业：人工智能

学号：58121102

年级：21 级大二

姓名：蒋雨初

签名：

时间：

实验一 数据降维与分类实验

1. 问题描述

概述

利用两种降维技术对葡萄酒数据进行降维，对降维前后的数据进行分类。

数据说明

所给数据集包含两个.csv 文件，分别为红葡萄酒数据(包括 1599 个样本)和白葡萄酒数据(包含 4898 个样本),每个样本含有 11 个特征(文件的前 11 列):固定酸度、挥发酸度、柠檬酸、残糖、氯化物、游离二氧化硫、总二氧化硫、密度、pH 值、硫酸盐、酒精和专家对此葡萄酒的打分(文件最后 1 列)。

实验内容

任务一: 对红白葡萄酒数据分别进行 PCA 和 LDA 降维处理，在降维后的数据集上完成基于 logistic 回归分类器的训练和测试。要求比较应用降维技术前后，分类器准确率的变化（对于降维后的数据，可以尝试利用可视化方法展示结果）。

任务二: 基于 MindSpore 平台提供的官方模型库，对相同的数据集进行训练，并与自己独立实现的算法对比结果（包括但不限于准确率、算法迭代收敛次数等指标)，并分析结果中出现差异的可能原因，给出使用 MindSpore 的心得和建议。

任务三: (加分项)使用 MindSpore 平台提供的相似任务数据集（例如，其他的分类任务数据集）测试自己独立实现的算法，并与 MindSpore 平台上的官方实现算法进行对比，进一步分析差异及其成因。

注：由于 MindSpore 并没有 PCA 和 LDA 算法，这是因为 MindSpore 对标 PyTorch 或 TensorFlow 这样的深度学习框架，而后者亦不包含 PCA、LDA 这样的机器学习算法。因此后续实验基于 sklearn 对比。

2. 实现步骤与流程

任务一:

1. 数据预处理

用 pandas 分别读取红白酒数据集，然后合并到同一个 DataFrame 中，可知有 6497 个样本，共 12 个 features（将专家打分也作为 feature 之一）。之后，再添加一个 label，用于标记红葡萄酒和白葡萄酒。

(1) 去重

很容易检测出原始数据中存在许多重复数据，而重复数据会影响样本的分布。有些算法对重复值并不敏感，例如朴素贝叶斯、SVM。而有些算法，例如神经网络、决策树、KNN 就会被影响。特别地，对于本实验中的线性回归，样本重复也是不应发生的，因为模型假定了样本之间是不一样的，也就是计算参数中有个自由度概念，通常 n 减 1。样本重复就减小了它，使之更容易显著。因此我们需要去重。重复数据去除后还剩 5320 条数据。

(2) 离群值去除

离群值是指与其他样本明显不同的极端观测值，可能是由于数据收集错误、测量误差或其他异常情况引起的。去除离群值的目的包括以下几点：

- 减少异常值对模型的影响：离群值可能对模型的参数估计和预测结果产生较大的影响。逻辑回归模型对极端值比较敏感，离群值可能导致模型产生偏差和不稳定的结果。通过去除离群值，可以减少这种影响，提高模型的稳定性和准确性。
- 改善特征的分布：离群值可能导致特征分布的偏斜或非正常形状，进而影响模型的学习能力。通过去除离群值，可以更好地恢复特征的正常分布，提高模型对数据的拟合能力。
- 提高模型的泛化能力：群值通常是不常见的、异常的数据点，它们可能无法很好地表示整体数据集的特征。当我们的目标是构建具有较好泛化能力的模型时，去除离群值可以减少模型对异常情况的过拟合，使其更好地适应一般情况下的数据。

这一步根据 IQR(Interquartile Range, 四分位距) 完成，我们只保留落于$[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$的数据。去除离群值之后还剩下 3671 条样本。

(3) 标准化

标准化是数据预处理中常用的一种方法，其目的是将不同特征的数据转换为具有统一尺度和范围的形式。在 PCA 算法中，强制要求数据进行标准化，下面是标准化的其他几个主要目的：

- 梯度下降的收敛速度：逻辑回归通常使用梯度下降算法来最小化损失函数并拟合模型。当特征具有不同的尺度或变化范围时，梯度下降的收敛速度可能会受到影响。通过对特征进行标准化，可以使特征具有相似的尺度，促进梯度下降算法的更快收敛。
- 模型解释性：在逻辑回归中，特征的系数（或权重）反映了其对预测结果的影响程度。当特征具有不同的尺度时，系数的大小也会受到尺度影响，难以比较不同特征的重要性。通过标准化特征，可以使得系数的大小可比较，提高模型的解释性。
- 避免特征偏向：如果某个特征具有较大的尺度，逻辑回归模型可能会偏向于该特征对预测结果的影响。这可能导致模型过度依赖这个特征，忽略其他特征的影响。通过标准化特征，可以避免特征偏向，使得模型更加平衡地考虑所有特征的影响。

具体公式如下,其中μ和σ分别为样本均值和标准差。

$$\tilde{x} = \frac{x - \mu}{\sigma}$$

2. 逻辑线性回归分类

将$z = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{b}$代入到 sigmoid 函数$y = 1/(1 + e^{-z})$中，再作简单变换得到

$$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{b} = \ln\frac{y}{1 - y}$$

令$p(y = 1|x) = y, p(y = 0|x) = 1 - y$，则得

$$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{b} = \ln\frac{p(y = 1|x)}{p(y = 0|x)}$$

与$p(y = 0|x) + p(y = 1|x) = 1$联立，解得

$$\begin{cases} p(y = 1|x) = \dfrac{e^{w^{\mathsf{T}}x+b}}{1 + e^{w^{\mathsf{T}}x+b}} \\ p(y = 0|x) = \dfrac{1}{1 + e^{w^{\mathsf{T}}x+b}} \end{cases}$$

为简化计算，我们在原特征矩阵$\boldsymbol{x}$的基础上新增一列全为 1 的特征，生成新的特征矩阵$\boldsymbol{X} = (\boldsymbol{x}; \boldsymbol{1})$；同时截距$\boldsymbol{b}$合并入$\boldsymbol{w}$，生成新的系数矩阵$\mathbf{B} = (\boldsymbol{w}; \boldsymbol{b})$。使

用极大似然估计得到对数似然函数

$$L(\boldsymbol{B}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} y_i \ln p(y=1|\boldsymbol{X}_i) + (1-y_i)\ln p(y=0|\boldsymbol{X}_i)$$

$$= \sum_{\mathrm{i}=1}^{\mathrm{n}} y_i \boldsymbol{B}^{\top}\boldsymbol{X}_i - \ln(1+e^{\boldsymbol{B}^{\top}\boldsymbol{X}_i})$$

求得其一阶导数为

$$\frac{\partial L}{\partial \boldsymbol{B}} = -\sum_{i=1}^{n} \boldsymbol{X}_i(y_i - \hat{y}_i)$$

并运用梯度下降法，设置学习率并迭代适宜轮数。

以上是二分类问题，所以使用 sigmoid 作为激活函数，如果要拓展到多分类问题，有以下两种常见做法：

- 一对多（One-vs-Rest，OvR）策略：对于多类别问题，我们将每个类别分别与其他类别进行比较。对于每个类别，训练一个二分类逻辑回归模型，将其作为"正类"，将其他所有类别作为"负类"。这样我们就得到了多个二分类模型，每个模型用于区分一个类别与其他类别的情况。在预测时，将测试样本传递给所有模型，选择输出概率最高的类别作为预测结果。
- 多项逻辑回归（Multinomial Logistic Regression）：多项逻辑回归模型是一种广义线性模型，用于处理多类别分类问题。它通过使用 softmax 函数，将线性模型的输出转换为多个类别的概率分布。在训练时，我们使用多类别的损失函数（如交叉熵）来最小化预测概率与真实标签之间的差异。在预测时，选择具有最高概率的类别作为预测结果。

本实验中我使用了后者。

3. 共线性检测

共线性（Collinearity）指的就是在拟合线性回归模型的过程中两个或两个以上的 predictors 互相之间存在较强的相关关系。当自变量之间存在共线性时，模型的参数会变得极其不稳定，模型得预测能力会下降。很难确切区分每个自变量对因变量的影响，因此增加了对于模型结果得解释成本。对此，可以

通过观察热力图来分析是否存在共线性。

绘制热力图，颜色越靠近红色则线性相关程度越大。从图中可以看出，残糖和密度、酒精读书和密度的线性相关性是较高的。

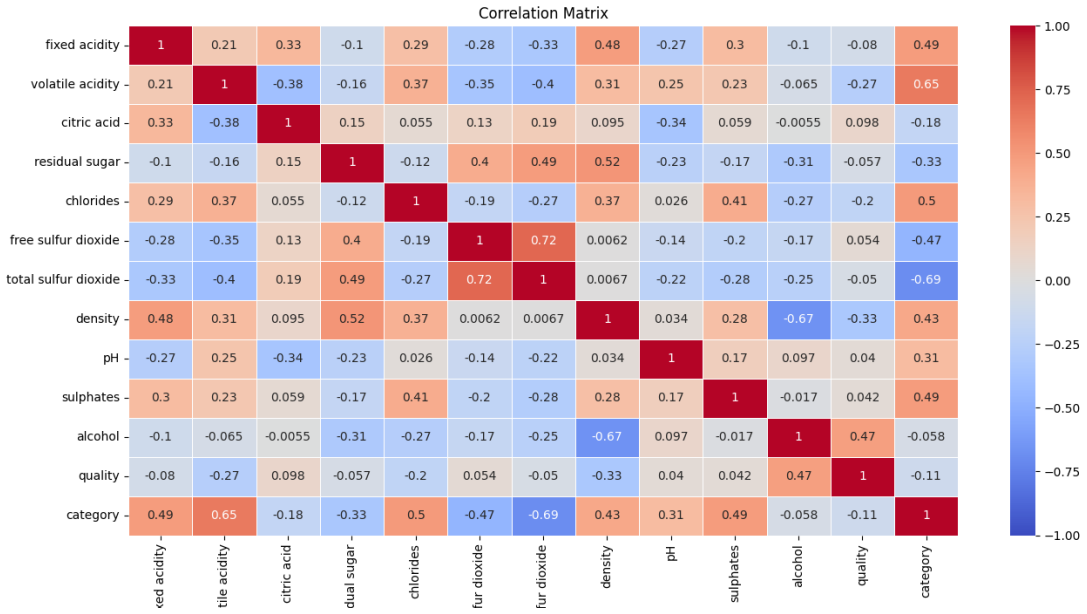

图 1 降维前的各特征线性相关热力图

当然，并不是所有的共线性都可以通过观察相似系数得到，还可以考虑计算方差膨胀因子（Variance Inflation Factor，VIF）。VIF 是指解释变量之间存在多重共线性时的方差与不存在 多重共线性时的方差之比。VIF 越大，显示共线性越严重。公式如下：

$$VIF_i = \frac{1}{1 - R_i^2}$$

其中， R_i为第 i 个变量X_i与其他全部变量$X_j(i = 1,2,\dots,k; i \neq j)$的复相关系数，所谓复相关系数即可决系数$R^2$的算术平方根，也即拟合优度的算术平方根。不过这个可决系数R_i^2是指用X_i做因变量，对其他全部X_j做一个新的回归以后得到的可决系数。当$VIF > 5$时我们认为有明显的共线性问题。

计算各个特征的 VIF，如图 2 。

feature	VIF
1	3.16
2	1.50
3	1.27
4	8.86
5	2.29
6	1.99
7	2.27
8	22.57
9	2.43
10	1.36
11	7.16
12	1.39

图 2 各个特征的方差膨胀因子

nth component	cumulative explained variance
1	25.00%
2	44.70%
3	57.11%
4	67.23%
5	74.48%
6	80.53%
7	85.76%
8	90.81%
9	94.64%
10	97.58%
11	99.79%
12	100.00%

图 3 第 n 个主成分的累计解释方差

从图 2 各个特征的方差膨胀因子中可以看出有 3 个特征的$VIF > 5$，因此存在共线性问题，可以使用 PCA 进行数据降维。

4. PCA 处理

m 维随机变量$\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^{\top}$的分量依次是$\boldsymbol{x}$的第一主成分到第 m 主成分的充要条件是：（1）$\boldsymbol{y} = \boldsymbol{A}^{\top}\boldsymbol{x}$，$\boldsymbol{A} = [\boldsymbol{\alpha}_1 \quad \boldsymbol{\alpha}_2 \quad \cdots \quad \boldsymbol{\alpha}_m]$为正交矩阵。（2）$cov(\boldsymbol{y}) = diag(\lambda_1, \lambda_2, \cdots, \lambda_m), \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$。其中$\lambda_k$是样本总体协方差$\boldsymbol{\Sigma}$的第 k 个特征值，$\boldsymbol{\alpha}_k$是对应的单位特征向量，即$\boldsymbol{\Sigma\alpha}_k = \lambda_k\boldsymbol{\alpha}_k$。用矩阵表示为$\boldsymbol{\Sigma A} = \boldsymbol{A\Lambda}$。

获得主成分随机变量$\boldsymbol{y}$后，再考察方差贡献率。第 k 主成分y_k的方差贡献率定义为y_k的方差和所有的方差之和的比，记作η_k：

$$\eta_k = \frac{\lambda_k}{\sum_{i=i}^{m}\lambda_i}$$

k 个主成分$y_1, y_2, \cdots, y_k$的累计方差贡献率定义为 k 个方差之和与所有方差之和的比：

$$\sum_{i=1}^{k}\eta_i = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=i}^{m}\lambda_i}$$

通常取 k 使得累积方差贡献率达到规定的百分比以上，如70%~80%以上。累积方差贡献率反映了主成分保留信息的比例，但它不能反应某个原有变量x_i保留信息的比例，这时通常利用 k 个主成分$y_1, y_2, \cdots, y_k$对原有变量x_i的贡献率。

传统 PCA 通过数据的协方差矩阵或相关矩阵的特征值分解进行，具体步骤如下：

(1) 对数据进行规范化处理，以$\boldsymbol{X}$表示。

(2) 计算样本相关矩阵$\boldsymbol{R}$:

$$\boldsymbol{R}=\left[\frac{1}{n-1}\sum_{l=1}^{n}x_{il}x_{lj}\right]_{\boldsymbol{m}\times\boldsymbol{m}}=\frac{1}{n-1}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$$

(3) 求相关矩阵$\boldsymbol{R}$的 k 个特征值和对应的 k 个单位特征向量，即求解特征方程

$$|\boldsymbol{R}-\lambda\boldsymbol{I}|=0$$

得$\boldsymbol{R}$的 m 个特征值

$$\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_m$$

求方差贡献率$\sum_{i=1}^{k}\eta_i$达到预定值得主成分个数 k。求前 k 个特征值对应的单位特征向量$\boldsymbol{\alpha}_i=(a_{1i},a_{2i},\cdots,a_{mi})^{\mathrm{T}}$。

(4) 求 k 个样本主成分。以 k 个单位特征向量为系数进行线性变换，求出 k 个样本主成分$y_i=\boldsymbol{\alpha}_i^{\mathrm{T}}\boldsymbol{x}$。

(5) 计算各个主成分的个体解释方差与累计解释方差，如图 4。设定阈值为 95%，从图 3 第 n 个主成分的累计解释方差知，应保留 9 个主成分。

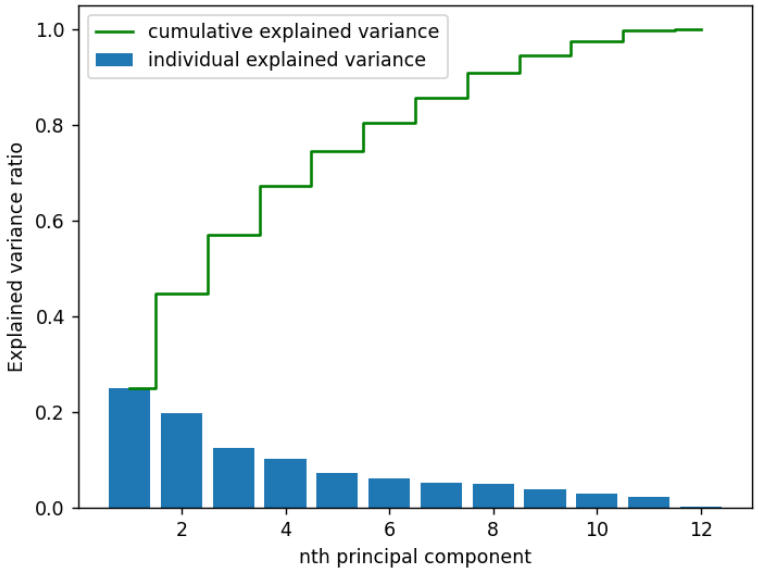

图 4 各个主成分的个体解释方差与累计解释方差

进行 PCA 降维之后，重新绘制热力图，从图 5 可见，共线性问题已得到较好的解决。

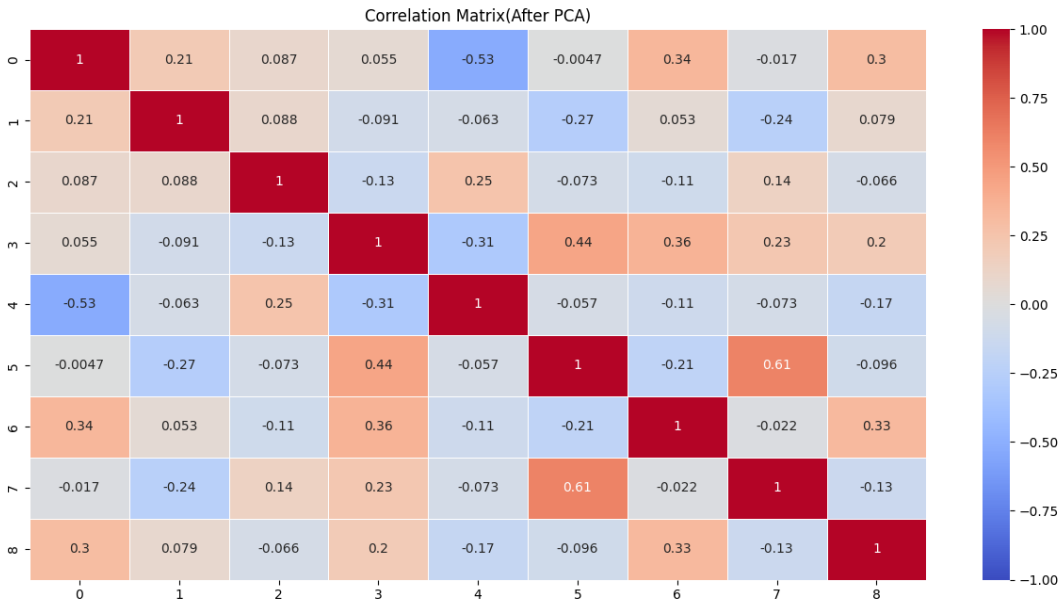

图 5 降维后的线性相关热力图

重新进行逻辑回归，并记录结果。

5. LDA 处理

不同于 PCA 方差最大化理论，LDA 算法的思想是将数据投影到低维空间之后，使得同类数据尽可能的紧凑（为此可以让同类样例投影点的协方差尽可能的小，即$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w}+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}$尽可能小。其中$\boldsymbol{w}$是投影方向，$\boldsymbol{\Sigma}_i$表示第$i\in\{0,1\}$表示例的协方差矩阵），不同类的数据尽可能分散（为此可以类与类中心距离尽可能大，即$\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2$ 尽可能大）。同时考虑二者，则可得到欲最大化的目标

$$\begin{aligned} J &= \frac{\|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_0-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1\|_2^2}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_0\boldsymbol{w}+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}} \\ &= \frac{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\Sigma}_0+\boldsymbol{\Sigma}_1)\boldsymbol{w}} \end{aligned} \tag{1}$$

定义类内散度矩阵$\boldsymbol{S}_w=\boldsymbol{\Sigma}_0+\boldsymbol{\Sigma}_1$和类间散度矩阵$\boldsymbol{S}_b=(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)^{\mathrm{T}}$代入到(1)式，得到

$$J=\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w}} \tag{2}$$

同时，LDA 有如下两个假设:

(1) 原始数据根据样本均值进行分类。

(2) 不同类的数据拥有相同的协方差矩阵。

于是可令$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w}=1$，则优化目标为

$$\min_{\boldsymbol{w}} -\boldsymbol{w}^{\top}\boldsymbol{S}_b\boldsymbol{w} \tag{3}$$

$$s.t.\ \boldsymbol{w}^{\top}\boldsymbol{S}_w\boldsymbol{w} = 1 \tag{4}$$

运用拉格朗日乘子法，得到

$$\boldsymbol{S}_b\boldsymbol{w} = \lambda\boldsymbol{S}_w\boldsymbol{w} \tag{5}$$

其中$\boldsymbol{\lambda}$是拉格朗日乘子。对(5)式作简单变化可以得到

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_b\boldsymbol{w} = \lambda\boldsymbol{w} \tag{6}$$

对(6)式进行特征值分解即$\boldsymbol{S}_w^{-1}\boldsymbol{S}_b = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$，其中 **Q** 是 *N*×*N* 方阵，且其第 *i* 列为 **A** 的特征向量。**Λ** 是对角矩阵，其对角线上的元素为对应的特征值。

事实上，如果是二分类问题也可以使用奇异值分解。注意到$\boldsymbol{S}_b\boldsymbol{w} = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\top}\boldsymbol{w}$的方向恒为$(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$（这是因为$(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\top}\boldsymbol{w}$是标量），所以可以设$(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\top}\boldsymbol{w} = \gamma$。代入到(5)式得到

$$\gamma(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) = \lambda\boldsymbol{S}_w\boldsymbol{w} \tag{7}$$

$$\boldsymbol{w} = \frac{\gamma}{\lambda}\boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) \tag{8}$$

由于最终要求解的$\boldsymbol{w}$不关心其大小，只关心其方向，所以其大小可以任意取值。由于$\boldsymbol{\mu}_0$和$\boldsymbol{\mu}_1$的大小是固定的，所以γ的大小只受$\boldsymbol{w}$的大小影响，因此可以调整$\boldsymbol{w}$的大小使得$\gamma = \lambda$。最终

$$\boldsymbol{w} = \boldsymbol{S}_w^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) \tag{9}$$

对$\boldsymbol{S}_w$进行奇异值分解，即$\boldsymbol{S}_w = \boldsymbol{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^{\top}$，这里$\boldsymbol{\Sigma}$是一个实对角矩阵，其对角线上的元素是$\boldsymbol{S}_w$的奇异值。再由$\boldsymbol{S}_w^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\top}$计算出$\boldsymbol{w}$。

现在把 LDA 推广到多分类任务中。假定存在 N 个类，首先定义全局散度矩阵

$$\boldsymbol{S}_t = \boldsymbol{S}_w + \boldsymbol{S}_b = \sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^{\top} \tag{10}$$

其中$\boldsymbol{\mu}$是所有实例的均值向量。将类内散度矩阵重定义为每个类别的散度矩阵之和，即

$$\boldsymbol{S}_w = \sum_{i=1}^{N}\boldsymbol{S}_{wi} = \sum_{i=1}^{N}\sum_{\boldsymbol{x}\in X_i}(\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^{\top} \tag{11}$$

由(10)(11)得重定义的类间散度矩阵$\boldsymbol{S}_b$

$$\boldsymbol{S}_b = \boldsymbol{S}_t - \boldsymbol{S}_w = \sum_{i=1}^{N} m_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top \tag{12}$$

其中m_i是第 i 类样本数量。同二分类，可以得到优化目标为

$$\min_{\boldsymbol{W}} -tr(\boldsymbol{W}^\top \boldsymbol{S}_b \boldsymbol{W}) \tag{13}$$

$$s.t.\ \ tr(\boldsymbol{W}^\top \boldsymbol{S}_w \boldsymbol{W}) = 1 \tag{14}$$

只需通过以下广义特征值问题求解

$$\boldsymbol{S}_b \boldsymbol{W} = \lambda \boldsymbol{S}_w \boldsymbol{W} \tag{15}$$

事实上，多分类任务亦可以使用奇异值分解来求解，但以下不展开讨论。

任务二:

1. 数据预处理

处理同任务一。

2. 使用 sklearn

直接使用 sklearn 中的 PCA 和 LinearDiscriminantAnalysis 进行数据降维，再使用自己的线性逻辑回归模型进行分类，记录训练损失、测试机准确率等指标。

任务三:

使用 MNIST 手写数字数据集合进行分类，并比较自己实现的算法和 sklearn 中的算法的性能差别。

3. 实验结果与分析

任务一

1. 数据降维

分别绘制原始数据、数据清洗后的原始数据、经过 PCA 降维后的数据和经过 LDA 降维后的数据，如图 6。

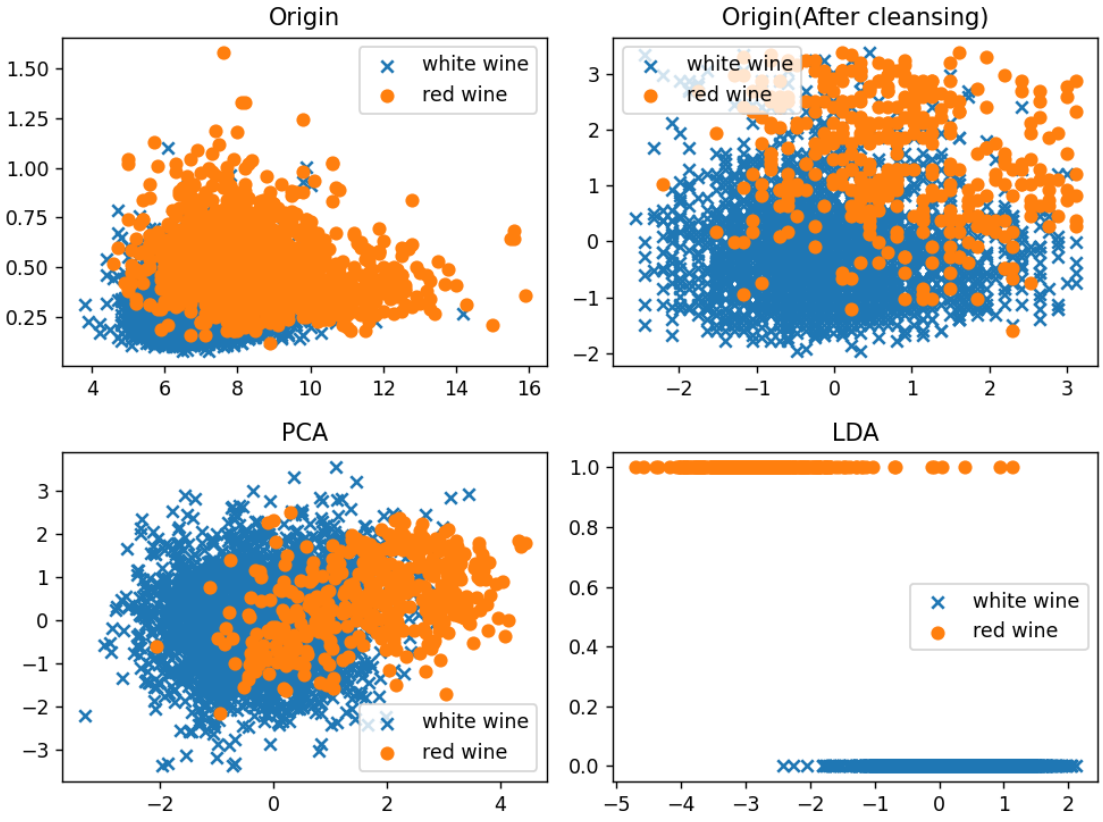

图 6 使用 PCA 或 LDA 降维后的数据情况

2. 分类任务

在四个数据集上分别进行训练，绘制 train loss 随训练轮数变化的情况，如图 7，可以发现如果直接在原始数据集进行训练，那么训练损失的收敛会非常不稳定。再从图 8 中可以看到，使用 LDA 进行降维的分类效果是最佳的。

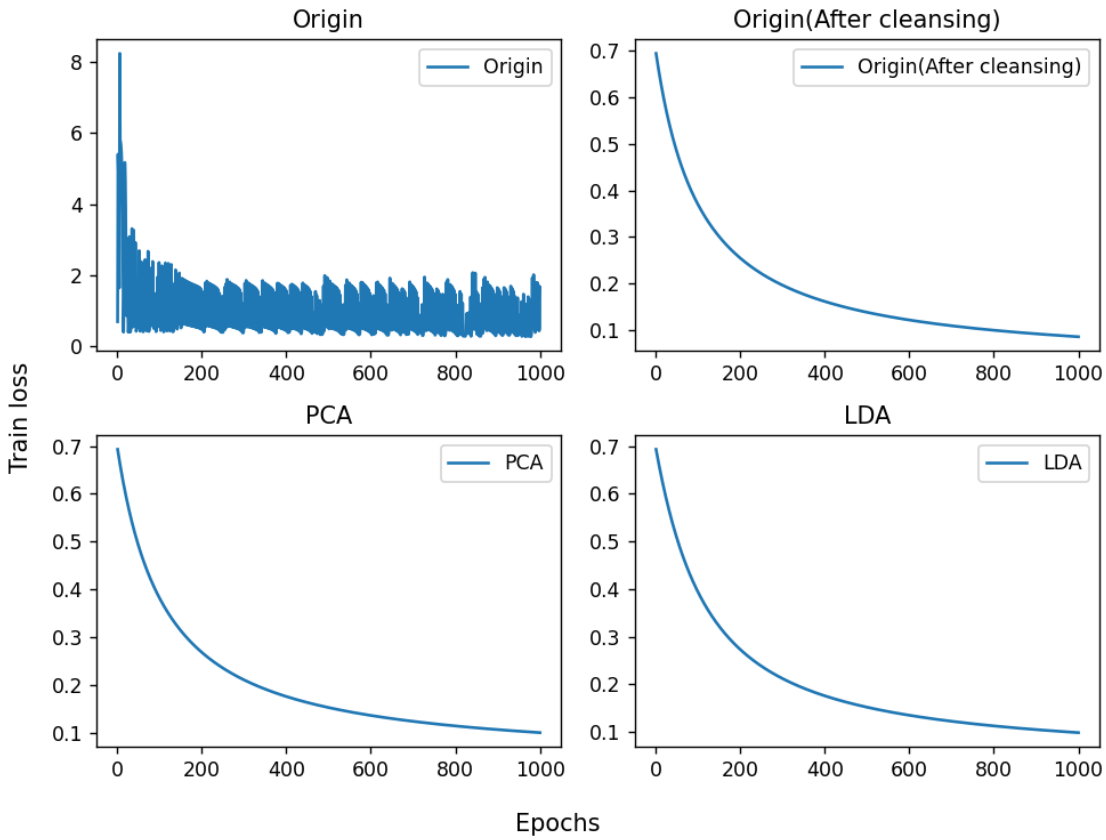

图 7 在不同处理方式的训练集上训练的训练损失随轮数的变化曲线

```
train on Origin, accuracy : 0.9342105263157895
train on Origin(After cleansing), accuracy : 0.9904632152588556
train on PCA, accuracy : 0.9918256130790191
train on LDA, accuracy : 0.9918256130790191
```

图 8 在不同处理方式的数据集上的最终分类准确率

任务二

1. 数据降维

分别绘制使用自己实现的算法和 sklearn 中算法在数据集上降维的情况。通过对比，可以发现，sklearn 的算法可以把两种数据分的更开。在 PCA 中这有可能是 sklearn 使用了奇异值分解带来了更好的数值稳定性。

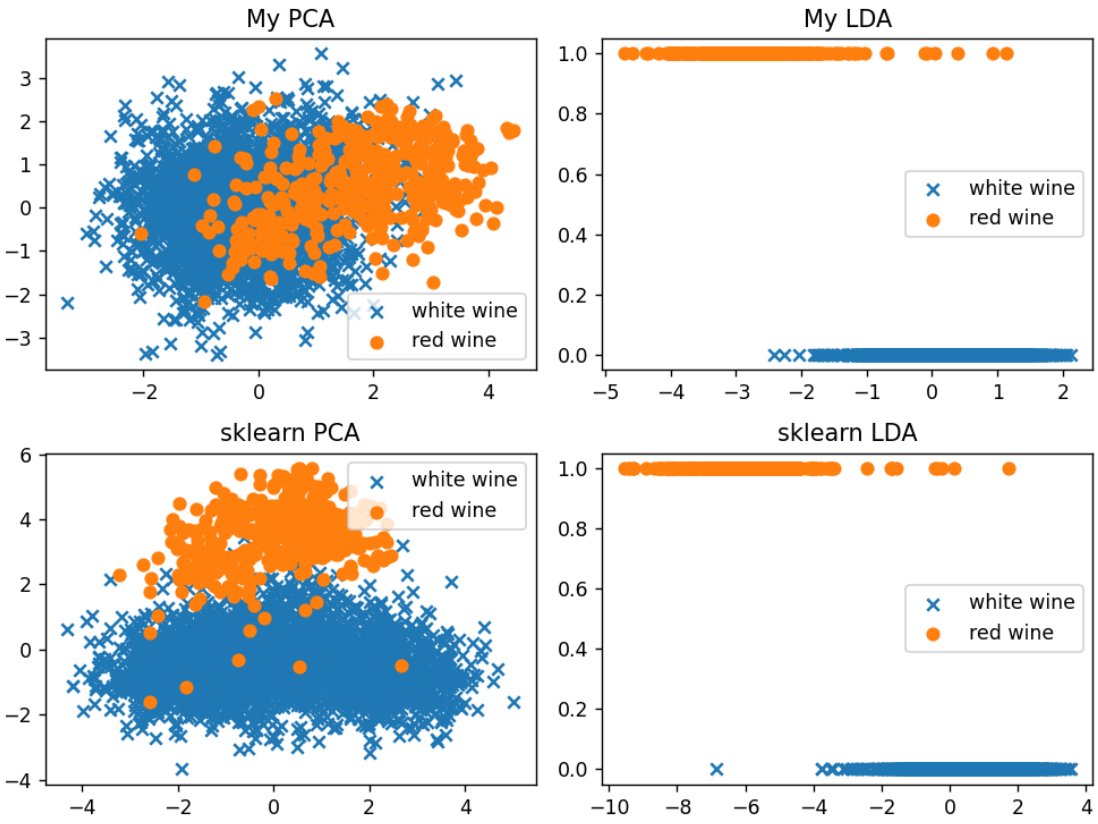

图 9 使用自己实现的算法和 sklearn 的算法对数据集的降维情况

2. 分类任务

分别使用自己实现和 sklearn 实现的 PCA、LDA 算法对数据集降维然后使用线性回归模型分类，绘制 train loss 随训练轮数变化的情况，如图 10。可以看到训练损失非常接近。

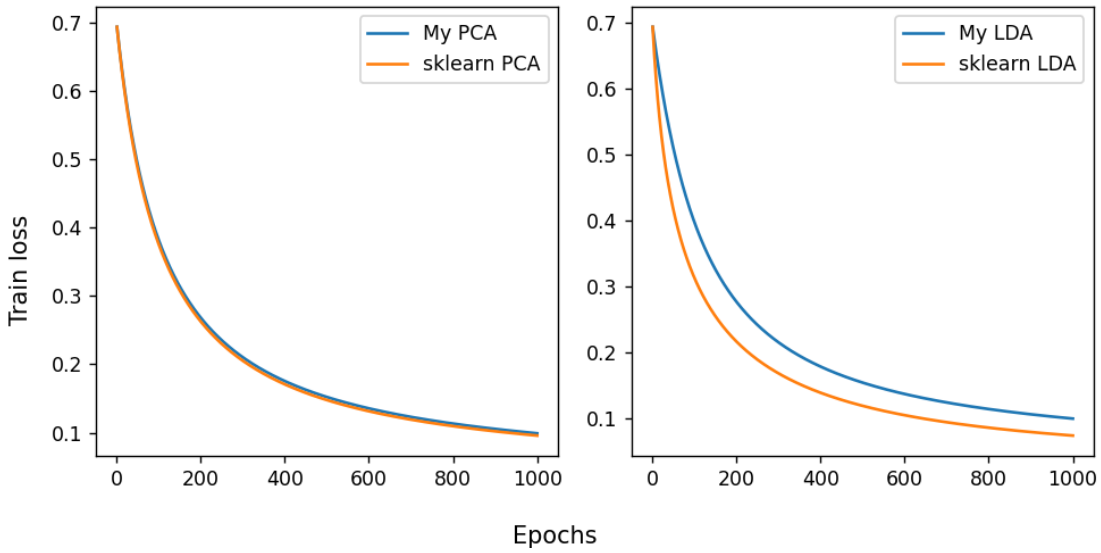

图 10 分别使用自己实现和 sklearn 实现的 PCA、LDA 算法在数据集上训练的损失随轮数变化曲线

```
train on My PCA, accuracy : 0.9891067538126361
train on My LDA, accuracy : 0.9912854030501089
train on sklearn PCA, accuracy : 0.9891067538126361
train on sklearn LDA, accuracy : 0.9956427015250545
```

图 11 分别使用自己实现和 sklearn 实现的 PCA、LDA 算法在数据集上的最终分类准确率

任务三

1. 数据降维

分别绘制使用自己实现的算法和 sklearn 中算法在 MNIST 上降维的情况。通过对比，可以发现，sklearn 的 PCA 算法可以把数据分的更开，而 LDA 算法二者差别较小。

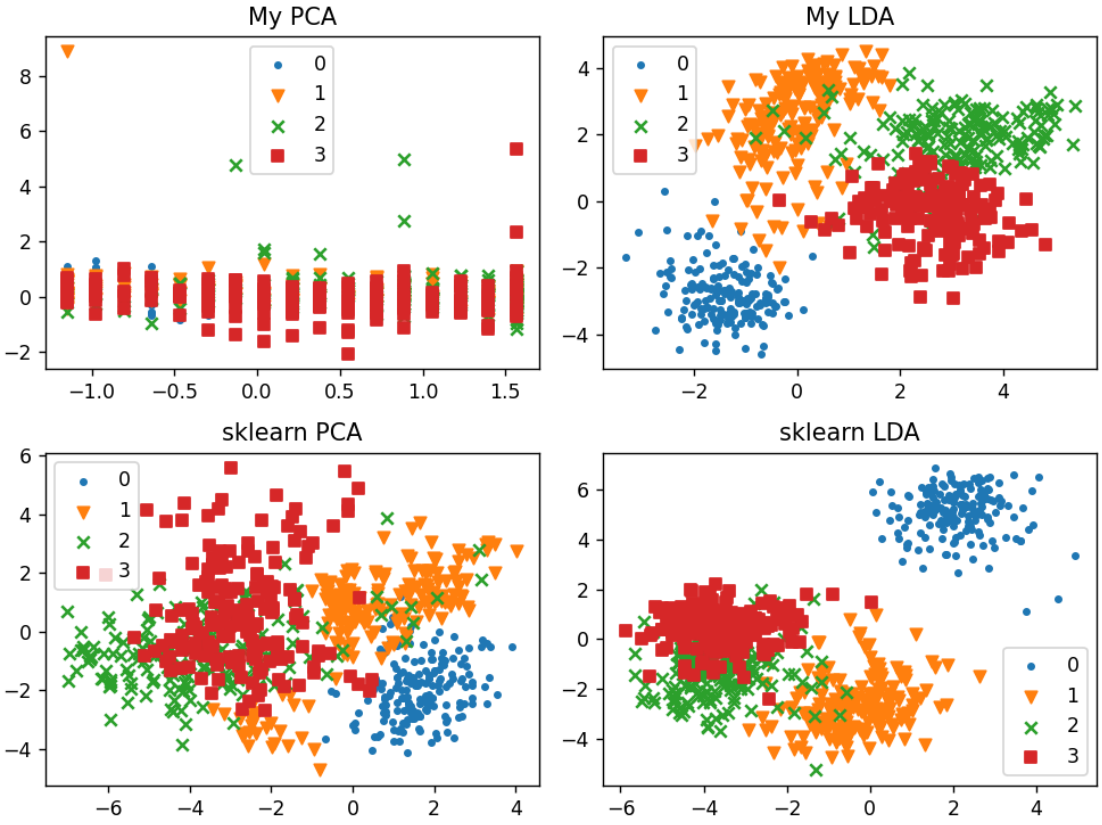

图 12 使用自己实现的算法和 sklearn 的算法对 MNIST 数据集的降维情况（只显示 0~3）

2. 分类任务

分别使用自己实现和 sklearn 实现的 PCA、LDA 算法对数据集降维然后使用线性回归模型分类，绘制 train loss 随训练轮数变化的情况。在多分类任务中，sklearn 的优势就被放大了出来，两项算法都要优于自己实现的。

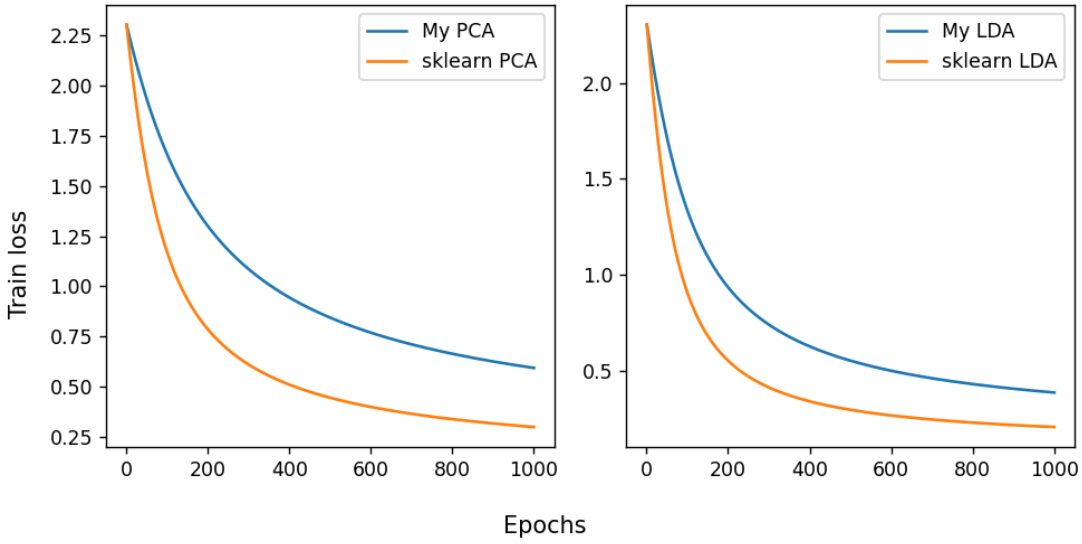

图 13 使用自己实现的算法和 sklearn 的算法对 MNIST 数据集进行分类任务的训练损失与轮数曲线

```
train on My PCA, accuracy : 0.88
train on My LDA, accuracy : 0.9088888888888889
train on sklearn PCA, accuracy : 0.9244444444444444
train on sklearn LDA, accuracy : 0.9666666666666667
```

图 14 使用自己实现的算法和 sklearn 的算法对 MNIST 数据集的最终分类准确率

实验二　KNN 分类任务实验

1. 问题描述

概述

利用 KNN 算法对 Iris 鸢尾花数据集中的测试集进行分类。

数据说明

鸢尾花数据集内包含的 3 类分别为山鸢尾（Iris-setosa）、变色鸢尾（Iris- versicolor）和维吉尼亚鸢尾（Iris-virginica），共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：花萼长度、花萼宽度、花瓣长度、花瓣宽度。标签 0、1、2 分别表示山鸢尾、变色鸢尾、维吉尼亚鸢尾。

数据集已被划分为训练集、验证集和测试集，分别存储于 data 文件夹中的 train.csv, val.csv, test.csv。train.csv 和 val.csv 文件包含 data, label 字段，分别存储着特征$X \in R^{N\times d}$和标记$Y \in R^{N\times 1}$。其中，N是样例数量，$d = 4$为特征维度，每个样例的标记$y \in \{0,1,2\}$。test.csv 文件仅包含 data 字段。

实验内容

任务一：利用欧式距离作为 KNN 算法的度量函数对测试集进行分类。在验证集上分析近邻数 K 对 KNN 算法分类精度的影响。

任务二：利用马氏距离作为 KNN 算法的度量函数，对测试集进行分类。在马氏距离中M为半正定矩阵，正交基A使得$M = AA^{\top}$成立。给定以下目标函数，在训练集上利用梯度下降法对马氏距离进行学习：

$$f(\mathbf{A}) = \min_{\mathrm{A}} \sum_{i=1}^{N} \sum_{j\in\Omega_i} p_{ij}$$

其中，Ωi 表示与 Xi 属于相同类别的样本的下标的集合，p_{ij}定义为：

$$p_{ij} = \begin{cases} \dfrac{\exp\left(-d_M(x_i,x_j)^2\right)}{\sum_{k\neq i}\exp\left(-d_M(x_i,x_k)^2\right)} & j \neq i \\ 0 & j = i \end{cases},$$

d_M为：

$$d_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|_2 = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}^T \boldsymbol{A}(\boldsymbol{x}_i - \boldsymbol{x}_j)}$$

并给出$\frac{\partial f(A)}{\partial A}$推导。

任务三：基于 MindSpore 平台提供的官方模型库，对相同的数据集进行训练，并与自己独立实现的算法对比结果（包括但不限于准确率、算法迭代收敛次数等指标)，并分析结果中出现差异的可能原因，给出使用 MindSpore 的心得和建议。

任务四：(加分项)使用 MindSpore 平台提供的相似任务数据集（例如，其他的分类任务数据集）测试自己独立实现的算法，并与 MindSpore 平台上的官方实现算法进行对比，进一步分析差异及其成因。

注：由于 MindSpore 并没有 KNN 算法，这是因为 MindSpore 对标 PyTorch 或 TensorFlow 这样的深度学习框架，而后者亦不包含 KNN 这样的机器学习算法。因此后续实验基于 sklearn 对比。

2. 实现步骤与流程

任务一

1. 数据预处理

(1) 去重

重复样本对 KNN 算法有一定的影响，尤其是在使用欧氏距离或其他基于距离的度量方法时。对于 KNN 算法中的每个最近邻样本，它们的权重是根据距离计算的。如果训练集中存在重复样本，那么这些重复样本将在最近邻中占据多个位置，从而影响了它们的权重。重复样本的存在可能导致某些样本在投票过程中具有更大的权重，从而对最终的分类结果产生影响。

综上，在进行算法事件之前需要先检查是否有重复数据，而在本次实验中并没有重复数值。

(2) 数据标准化

标准化是数据预处理中常用的一种方法，其目的是将不同特征的数据转换

为具有统一尺度和范围的形式。

具体公式如下,其中μ和σ分别为样本均值和标准差。

$$\tilde{x} = \frac{x - \mu}{\sigma}$$

2. 距离计算

本任务中要求使用的距离度量为欧氏距离。

欧氏距离定义如下：

$$d_E(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{\sum_i (x_i - y_i)^2}$$

欧氏距离是最常见和直观的距离度量方法，它适用于特征空间是连续的情况。但是它对异常值敏感，容易受到噪声和特征尺度的影响，因此适用于特征之间存在线性关系且特征尺度相似的情况，如图像处理、文本分类等。

3. KNN 算法纲要

输入：训练样本集

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

输出：实例x所属的类y

(1) 根据给定的距离度量，在训练集中找出与x最邻近的 k 个点，涵盖这 k 个点的x的邻域记作$N_k(x)$。

(2) 在$N_k(x)$中根据分类决策规则（如多数表决）决定x所属的类y：

$$y = \arg\max_c \sum_{x_i \in N(x)} I(y_i = c)$$

其中I为指示函数。

4. KNN 实现：KD 树

实现 KNN 的关键在于如何对训练数据进行快速的 k 近邻搜索。实现 k 邻近最简单的方法是线性扫描，在本实验中由于数据较小，所以可以适用。当训练集很大时，计算非常耗时，这种做法是不可行的。

为了提高搜索效率，可以使用 kd 树。使用 kd 树的主要目的是加速最近邻搜索，尤其是在高维数据集上。它通过将数据点组织成一种树状结构，使得在搜索最近邻时可以快速剪枝和减少搜索空间。

kd 树（k-dimensional tree）是一种用于组织和搜索 k 维空间中数据点的数据

结构。它基于二叉树的概念，通过递归地划分空间并构建树结构来实现快速的最近邻搜索。

下面给出构造和搜索的算法。

构建树：

输入：k 维空间数据集$T = \{x_1, x_2, \dots, x_N\}$，其中$x_i = \{x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(k)}\}$

输出：kd 树

（1） 开始：构造根节点，根节点对应于包含T的 k 维空间的超矩形区域。

选择$x^{(1)}$为坐标轴，以T中所有实例的$x^{(1)}$的中位数为切分点，将根节点对应的超矩形区域切分为两个子区域。切分由通过切分点并与坐标轴$x^{(1)}$垂直的超平面实现。

由根节点生成深度为 1 的左右子节点：左节点对应坐标$x^{(1)}$小于切分点的子区域，右节点对应坐标$x^{(1)}$大于切分点的子区域。

（2） 重复：对深度为 j 的节点，选择$x^{(l)}$为切分的坐标轴，$l = j(mod\ k) + 1$，以该节点的区域中的所有实例的$x^{(l)}$坐标的中位数作为切分点，将该节点对应的超矩形区域切分为两个子区域。左右子树所对应区域同上。

（3） 直到两个子区域没有实例时停止划分

最近邻搜索：

输入：已构造的 kd 树，目标点x

输出：x 的最近邻

（1） 在 kd 树中找出包含目标点x的叶节点：从根结点出发，递归向下访问 kd 树。若目标点x当前维的坐标小于切分点的坐标，则进入左子树，否则进入右子树，直到子节点为叶子节点为止。

（2） 以此叶节点为“当前最近点”

（3） 递归地向上回退，在每个节点进行以下操作：

（a） 如果该节点保存的实例点比当前最近点距离目标点更近，则以当前实例点作为“当前最近点”

（b） 当前最近点一定存在于该节点一个子节点对应的区域。检查该子

节点的父节点的另一子结点对应的区域是否有更近的点。具体可以检查另一子节点对应区域是否与以目标点为球心、以目标点与“当前最近点”间的距离为半径的超球体相交。如果相交，则移动到另一个子节点，然后递归地进行最近邻搜索，否则向上回退。

（4） 当回退到根节点时，搜索结束。最后的“当前最近点”就是 x 的最近邻点。

5. K 值的选择

由 KNN 原理我们可知，当 k 过小时，模型更加敏感，容易受到噪声和异常值的影响。这可能导致模型过度关注局部的数据特征，而忽略了整体的数据分布，易发生过拟合；而 k 过大时，模型更加平滑，容易忽略局部的数据特征。这可能导致模型无法捕捉到数据的细微变化和边界信息，从而导致预测性能较差，易发生欠拟合。

在应用中，k 一般取一个比较小的值，然后采用交叉验证法来选取最优的 k 值。选用 5 折交叉，分别选择 1~10 作为 k 值，并在验证集上评估准确率，从图 15 可以看到最优的 k 值应该选择 7.

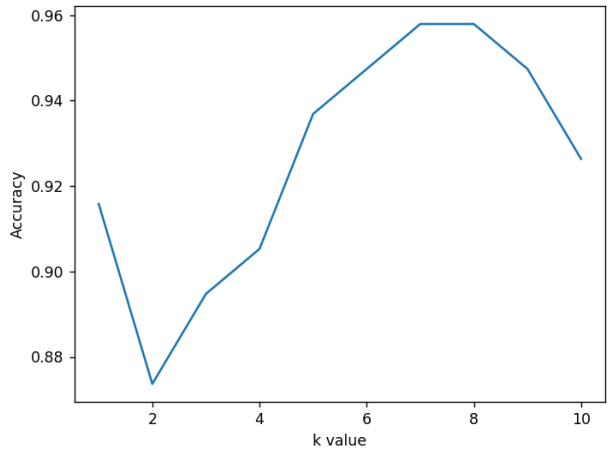

图 15 使用欧氏距离作为距离度量时不同 k 值下的分类准确率

任务二

1. 公式推导

$\frac{\partial f(A)}{\partial A}$推导如下：

记$g_{ij} = \exp\left(-\left\|Ax_i - Ax_j\right\|_2^2\right)$，则

$$p_{ij} = \frac{g_{ij}}{\sum_{k \neq i} g_{ik}}$$

那么

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

则

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

前半部分为：

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

$$= -2p_{ij}A(x_i - x_j)(x_i - x_j)^\top$$

后半部分为：

$$\frac{1}{(\sum_{k \neq i} g_{ik})^2} g_{ij}\sum_{k \neq i}\frac{\partial g_{ik}}{\partial A} = p_{ij}\frac{-2\sum_{k \neq i} g_{ik}A(x_i - x_k)(x_i - x_k)^\top}{\sum_{s \neq i} g_{is}}$$

$$= -2p_{ij}A\left(\sum_{k \neq i} p_{ik}\,(x_i - x_k)(x_i - x_k)^\top\right)$$

所以

$$\frac{\partial p_{ij}}{\partial A} = -2p_{ij}A\left((x_i - x_j)(x_i - x_j)^\top - \sum_{k \neq i} p_{ik}\,(x_i - x_k)(x_i - x_k)^\top\right)$$

最终

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

$$= -2A\sum_{i=1}^{N}\sum_{j \in \Omega_i} p_{ij}\left((x_i - x_j)(x_i - x_j)^\top - \sum_{k \neq i,j} p_{ik}\,(x_i - x_k)(x_i - x_k)^\top\right)$$

$$= 2A\sum_{i}\left(p_i\sum_{k \neq i,j} p_{ik}\,(x_i - x_k)(x_i - x_k)^\top - \sum_{j \in \Omega_i} p_{ij}\,(x_i - x_j)(x_i - x_j)^\top\right)$$

其中$p_i = \sum_{j \in \Omega_i} p_{ij}$，进一步化简

$$\frac{\partial f}{\partial A} = -2\sum_{i}\sum_{j} W_{ij}(Ax_i - Ax_j)(x_i - x_j)^\top$$

其中

$$W_{ij} = \begin{cases} p_{ij} & \text{if } j \in \Omega_i \\ 0 & \text{if } j \notin \Omega_i \end{cases}$$

进一步

$$\frac{\partial f}{\partial A} = 2\sum_i \sum_j W_{ij}(Ax_i x_i^\top + Ax_j x_j^\top - Ax_i x_j^\top - Ax_j x_i^\top)$$

$$= 2(XA^\top)^\top\big(diag\big(sum(W, axis = 0)\big) - W - W^\top\big)X$$

2. 数据预处理

同任务一。

3. 距离计算

本任务中要求使用的距离度量为马氏距离。

马氏距离可以看作是欧氏距离的一种修正，修正了欧式距离中各个维度尺度不一致且相关的问题，单个样本的马氏距离定义如下：

$$d_M(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})} \tag{1}$$

两个样本的马氏距离定义如下：

$$d_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{y})} \tag{2}$$

其中$\boldsymbol{\Sigma}$是多维随机变量的协方差矩阵，μ为样本均值，如果协方差矩阵是单位矩阵，也就是各维度独立同分布，马氏距离就变成了欧氏距离。所以，具体来说，我们要让样本维度间相互独立，再进行标准化。

由主成分分析可知，主成分就是特征向量方向，每个方向的方差就是对应的特征值，所以需要把样本按照特征向量的方向旋转，然后缩放特征值倍。

假设有样本$\boldsymbol{x}$，先来推导(1)式。对$\boldsymbol{x}$做正交变换得到$\hat{\boldsymbol{x}}$：

$$\hat{\boldsymbol{x}} = \boldsymbol{A}^\top \boldsymbol{x} \quad \hat{\boldsymbol{\mu}} = \boldsymbol{A}^\top \boldsymbol{\mu}$$

其中 $\boldsymbol{\mu}, \hat{\boldsymbol{\mu}}$为 $\boldsymbol{x}, \hat{\boldsymbol{x}}$的均值向量， $\boldsymbol{A} = [\boldsymbol{\alpha}_1 \quad \boldsymbol{\alpha}_2 \quad \cdots \quad \boldsymbol{\alpha}_m]$为正交矩阵。那么$\hat{\boldsymbol{x}}$的协方差矩阵$\hat{\boldsymbol{\Sigma}}$为：

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}(\hat{\boldsymbol{x}} - \hat{\boldsymbol{\mu}})(\hat{\boldsymbol{x}} - \hat{\boldsymbol{\mu}})^\top = \boldsymbol{A}^\top \boldsymbol{\Sigma} \boldsymbol{A}$$

其中$\boldsymbol{\Sigma}$为 $\boldsymbol{x}$的协方差矩阵。再对$\hat{\boldsymbol{x}}$做标准化代入到欧氏距离公式中得到d_M:

$$\left\| \frac{\hat{x} - \hat{\mu}}{\sqrt{\hat{\boldsymbol{\Sigma}}}} \right\|_2 = \|(\boldsymbol{A}^\top \boldsymbol{\Sigma} \boldsymbol{A})^{-\frac{1}{2}} \boldsymbol{A}^\top (\boldsymbol{x} - \boldsymbol{\mu})\|_2$$

$$= \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^{\top}\boldsymbol{A}(\boldsymbol{A}^{\top}\boldsymbol{\Sigma}\boldsymbol{A})^{-\frac{1}{2}}(\boldsymbol{A}^{\top}\boldsymbol{\Sigma}\boldsymbol{A})^{-\frac{1}{2}}\boldsymbol{A}^{\top}(\boldsymbol{x}-\boldsymbol{\mu})}$$

$$= \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^{\top}\boldsymbol{A}(\boldsymbol{A}^{\top}\boldsymbol{\Sigma}\boldsymbol{A})^{-1}\boldsymbol{A}^{\top}(\boldsymbol{x}-\boldsymbol{\mu})}$$

$$= \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^{\top}\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^{-1}\boldsymbol{A}^{\top}(\boldsymbol{x}-\boldsymbol{\mu})} \qquad (\boldsymbol{A}^{-1}=\boldsymbol{A}^{\top})$$

$$= \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} = d_M(\boldsymbol{x})$$

同理可以推得（2）式。

尽管马氏距离修正了欧氏距离的一些缺陷，但是它要求协方差矩阵必须满秩（因为需要求逆矩阵），如果没有可以考虑先进行 PCA 降维。

4. K 值的选择

选用 5 折交叉，分别选择 1~10 作为 k 值，并在验证集上评估准确率，从图 15 可以看到最优的 k 值应该选择 4.

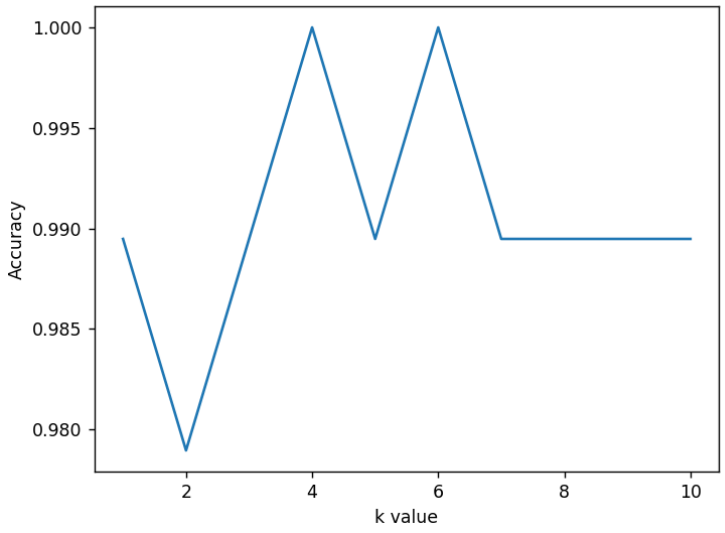

图 16 先在数据集上运用该算法，再进行 KNN 分类

任务三

在题目提供的数据集上进行分类任务，以欧氏距离和马氏距离作为距离度量，对比 sklearn 和自己实现的 KNN 的区别，并将在测试集上的预测结果写入到 task3_test_prediciton.csv 中。

任务四

在乳腺癌数据集上进行分类任务，分别以欧氏距离和马氏距离作为距离度

量，对比 sklearn 和自己实现的 KNN 的区别。

3. 实验结果与分析

任务一

使用前面选取出的最佳 k 值 7，最终在验证集上达到了 100%的准确率。

在测试集上进行预测，然后将结果写入到了 task1_test_prediction.csv 中

任务二

使用前面选取出的最佳 k 值 4，最终在验证集上达到了 93.33%的准确率。

在测试集上进行预测，然后将结果写入到了 task2_test_prediction.csv 中

任务三

使用 sklearn 的 KNN，可以看到最佳的 k 值是 8，与任务一中自己的实现相近，且最终在验证集上也达到了 100%的准确率。

在测试集上进行预测，然后将结果写入到了 task3_test_prediction.csv 中

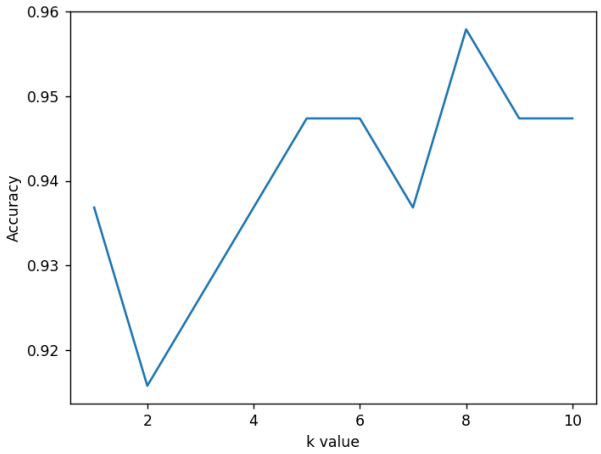

图 17 以欧氏距离作为性能度量时，sklearn 的 KNN 的表现

任务四

在乳腺癌数据集进行分类任务，如图 18 当使用欧氏距离时，在测试集上的准确率与 sklearn 的较为接近，但是使用马氏距离，在自己实现的 KNN 上表现较差，这有可能是因为：在我的实现中我使用了 KD 树，且使用了一种简单的方式来选择划分维度，即根据方差选择维度。这种方式可能无法很好地适应数据集的特征。而 sklearn 的算法是根据数据集自适应的，它会选择更好的算法来处理训练数据。当我强制指定 sklearn 的 KNN 使用 kd 树时甚至直接抛出了异常，这

更有力地证明了我的猜想。

```
Test Accuracy of sklearn KNN with Euclidean: 0.956140350877193
The best k of sklearn KNN with Euclidean: 8
Test Accuracy of sklearn KNN with Mahalanobis: 0.9385964912280702
The best k of sklearn KNN with Mahalanobis: 1
Test Accuracy of My KNN with Euclidean: 0.956140350877193
The best k of My KNN with Euclidean: 3
Test Accuracy of My KNN with Mahalanobis: 0.7631578947368421
The best k of My KNN with Mahalanobis: 1
```

图 18 分别在自己和 sklearn 的 KNN 上使用两种距离度量在乳腺癌数据集上进行测试的结果

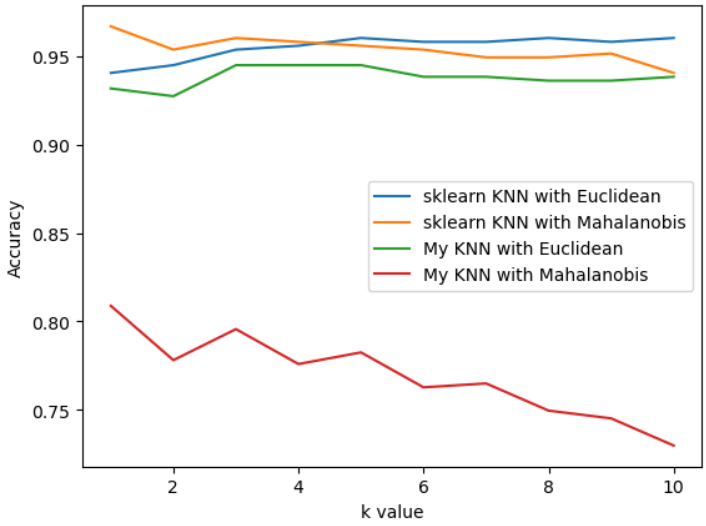

图 19 分别在自己和 sklearn 的 KNN 上使用两种距离度量在乳腺癌数据集上进行测试后绘制出的关于分类准确率和 k 值的曲线

实验三 神经网络

1. 问题描述

概述

利用神经网络算法，对 MNIST 数据集中给定的测试集进行分类。

数据描述

MNIST 是著名的手写体数字识别数据集。该数据集由训练数据集和测试数据集两部分组成，其中训练数据集包含了 60000 张样本图片及其对应标签，每张图片由 28×28 的像素点构成；测试数据集包含了 10000 张样本图片及其对应标签，每张图片由 28×28 的像素点构成。该数据集的训练数据可以通过 http://yann.lecun.com/exdb/mnist/地址下的下载链接下载，也可以自行搜索下载。

任务说明

任务一：基于神经网络模型及 BP 算法，根据训练集中的数据对所设计的神经网络模型进行训练，随后对给定的打乱的测试集中的数据进行分类。

任务二：基于 MindSpore 平台提供的官方模型库，对相同的数据集进行训练，并与自己独立实现的算法对比结果（包括但不限于准确率、算法迭代收敛次数等指标），并分析结果中出现差异的可能原因。

任务三：使用 MindSpore 平台提供的相似任务数据集测试自己独立实现的算法，并与 MindSpore 平台上的官方实现算法进行对比，进一步分析差异及其成因。

2. 实现步骤与流程

任务一：

1. 数据预处理

MNIST 数据集中的图像为灰度图，这意味着只有一个通道。且数值范围是[0,255]，因此需要标准化。先整体除以 255 将图片像素值压缩到[0,1]内，再运用如下公式,其中μ和σ分别为样本均值和标准差。

$$\tilde{x} = \frac{x - \mu}{\sigma}$$

2. 神经网络的构建

单隐层神经网络是一种基本的神经网络模型，它包含一个输入层、一个隐藏层和一个输出层。其中输入层的节点数为 784，对应于输入图像的 784 个像素；隐藏层的节点数为 256，它是一个可调整的参数，可以根据实际需求进行调整；输出层的节点数为 10，对应于 10 个不同的类别。

在单隐层神经网络中，每个节点都与前一层的所有节点相连，每个连接都有一个权重，它决定了该连接的重要性。隐藏层的节点通过激活函数（如 sigmoid 函数, 线性整流函数 ReLU）将输入信号加权求和后，输出一个非线性的结果。输出层的节点也通过激活函数（如 softmax 函数）将输入信号加权求和后，输出一个概率分布，表示输入图像属于每个类别的概率。

(1) 隐藏层结点数

测试了当结点数目为 128、256、512、1024、2048 时的情况，并绘制误差-轮数图像。由图中可以看出，最佳的隐藏层结点数应该为 1024。然而较大的神经网络训练会消耗更多的时间，且训练损失差异不大，所以仍然选择结点数为 256.

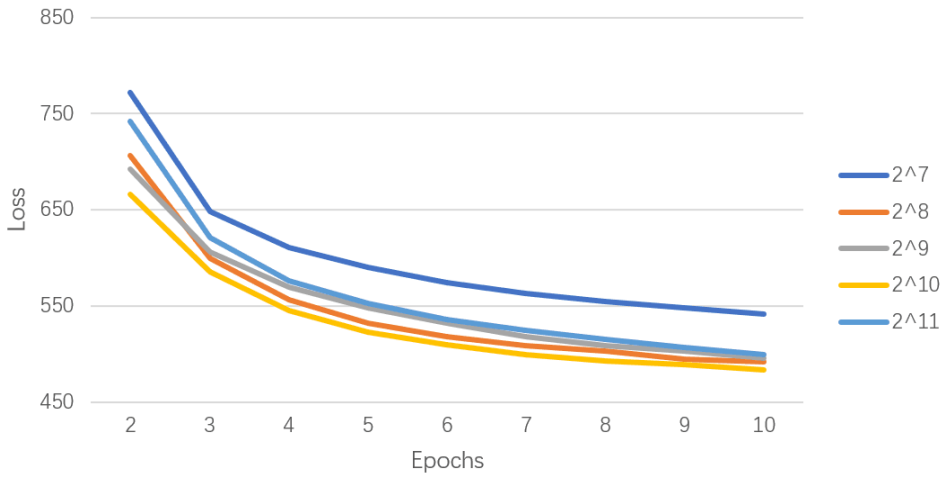

图 20 不同隐藏层结点数的测试集误差

3. 神经网络的训练

训练单隐层神经网络通常使用反向传播算法，该算法通过比较网络输出和实际标签之间的差异来调整权重，以最小化误差。在训练过程中，网络逐渐学习到输入数据的特征，从而提高分类的准确性。

算法概要如下：

正向传播，得到各层输出$\boldsymbol{h}^{(0)},\boldsymbol{h}^{(1)},\boldsymbol{h}^{(2)}$

$$\boldsymbol{z}^{(0)}=\boldsymbol{x},\boldsymbol{h}^{(0)}=\boldsymbol{z}^{(0)}$$

$$\boldsymbol{z}^{(1)}=\boldsymbol{W}^{(1)}\boldsymbol{h}^{(0)}+\boldsymbol{b}^{(1)},\boldsymbol{h}^{(1)}=relu\left(\boldsymbol{z}^{(1)}\right)$$

$$\boldsymbol{z}^{(2)}=\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)}+\boldsymbol{b}^{(2)},\boldsymbol{h}^{(2)}=softmax(\boldsymbol{z}^{(2)})$$

反向传播，得到输出层误差$\boldsymbol{\delta}^{(2)}$

$$\boldsymbol{\delta}^{(2)}=\frac{\partial L}{\partial \boldsymbol{z}^{(2)}}=\boldsymbol{h}^{(2)}-\boldsymbol{y}$$

其中 L 是损失函数。再计算各层梯度和隐藏层误差$\boldsymbol{\delta}^{(1)}$

$$\nabla_{\boldsymbol{W}^{(2)}}L=\boldsymbol{h}^{(2)^T}\cdot\boldsymbol{\delta}^{(2)}$$

$$\nabla_{\boldsymbol{b}^{(2)}}L=\boldsymbol{\delta}^{(2)}$$

$$\boldsymbol{\delta}^{(1)}=\frac{\partial relu\left(\boldsymbol{z}^{(1)}\right)}{\partial \boldsymbol{z}^{(1)}}\odot\left(\boldsymbol{\delta}^{(2)}\cdot\boldsymbol{W}^{(2)^T}\right)$$

$$\nabla_{\boldsymbol{W}^{(1)}}L=\boldsymbol{h}^{(1)^T}\cdot\boldsymbol{\delta}^{(1)}$$

$$\nabla_{\boldsymbol{b}^{(1)}}L=\boldsymbol{\delta}^{(1)}$$

其中 relu 的导数为

$$\frac{\partial relu(x)}{\partial x}=\begin{cases}1 & \text{if } x>0\\ 0 & \text{if } x\leq 0\end{cases}$$

更新各层参数

$$\boldsymbol{W}^{(t)}\leftarrow\boldsymbol{W}^{(t)}-\eta\nabla_{\boldsymbol{W}^{(t)}}L$$

$$\boldsymbol{b}^{(t)}\leftarrow\boldsymbol{b}^{(t)}-\eta\nabla_{\boldsymbol{b}^{(t)}}L$$

4. 超参数

学习率取 0.01，批大小取 64，篇幅受限不讨论这些参数的取法，下面着重讨论训练轮数 Epoch 的选择。

Epoch 是指将整个训练数据集在神经网络中前向传播和反向传播的一次完

整迭代。 Epoch 的数量直接影响模型是否能够收敛到最优解。较少的 Epoch 可能导致模型尚未完全学习训练数据中的模式和特征，从而无法达到最佳性能。较多的 Epoch 可以让模型有更多机会从训练数据中学习，但过多的 Epoch 可能导致过拟合，即模型过度适应训练数据而在未见过的数据上表现不佳。

绘制损失-训练周期图和准确率-训练周期图，从图 21 中可看出当尽管只训练了 10 轮但仍然已出现过拟合现象，因此需要使用早停法，选择在测试集上准确率最高的轮数，也就是 6 轮。

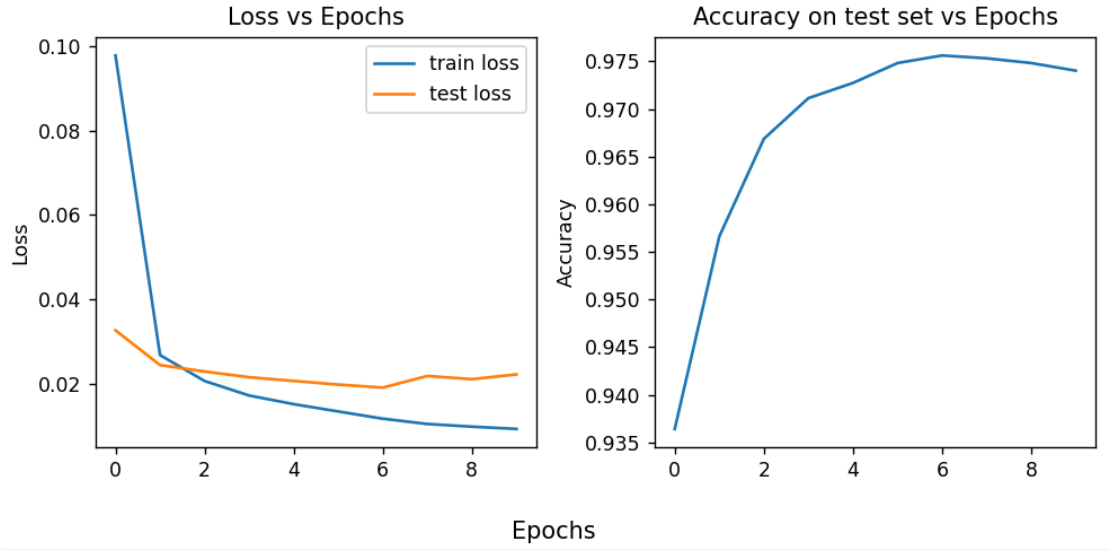

图 21 在手动实现的神经网络上训练 10 轮的模型收敛情况

5. 优化器

为了让模型更快地收敛，在本次实验中还手动实现了 adam 优化器。

Adam（Adaptive Moment Estimation）是一种自适应优化算法，用于训练机器学习和深度学习模型。它结合了动量法和自适应学习率的思想，旨在加快模型的收敛速度并提高优化性能。

Adam 算法维护了两个动量变量：一阶矩估计（即梯度的一阶矩）和二阶矩估计（即梯度的二阶矩）。这些动量变量类似于动量法中的动量向量，用于跟踪梯度的历史信息。

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective)
λ (weight decay), *amsgrad*, *maximize*
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ **to** ... **do**
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1-\beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

return θ_t

图 22 Adam 算法的伪代码

具体而言，Adam 算法在每个参数的维度上维护以下变量：

1. 梯度一阶矩估计（Mean of gradients）：表示过去梯度的平均值。

2. 梯度二阶矩估计（Variance of gradients）：表示过去梯度平方的平均值。

Adam 算法的参数更新规则可以表示为：

$$m = \beta_1 m + (1-\beta_1)\nabla J(\theta, x_i, y_i) \quad \text{（梯度一阶矩估计）}$$

$$v = \beta_2 v + (1-\beta_2)\big(\nabla J(\theta, x_i, y_i)\big)^2 \quad \text{（梯度二阶矩估计）}$$

m和v的初始值都为 0，β_1和β_2是动量衰减率（通常设置为接近 1 的值，例如 0.9 和 0.999）。

- 校正偏差（Bias Correction）：

由于 m 和 v 的初始值为 0，在训练的早期阶段，它们的估计会被偏向较低的值。为了解决这个问题，Adam 引入了校正偏差修正：

$$\hat{m} = \frac{m}{1 - \beta_1^t} \text{ (校正偏差修正)}$$

$$\hat{v} = \frac{v}{1-\beta_2^t}$$

其中，t 表示当前迭代的次数。

- 参数更新：

$$\theta = \theta - \frac{\alpha \hat{m}}{\sqrt{\hat{v}} + \varepsilon}$$

其中，α是学习率，ε是一个小的常数（如10^{-8}），用于避免除以零的情况。

Adam 算法的主要优势在于它结合了动量法和自适应学习率的优点。它能够自适应地调整每个参数的学习率，使得参数在梯度变化较大的方向上更快地更新，并在梯度变化较小的方向上缓慢更新。这有助于加快收敛速度，并且对于不同参数具有不同的学习率。

任务二:

1. 数据预处理

使用 MnistDataset 载入数据，然后用 dataset.map 对数据进行转换。主要包括标准化和将数据压缩到[0,1]，这部分与任务一相同。不同的地方在于还需要使用 mindspore.dataset.vision.HWC2CHW()对数据的维度进行重排，把通道数放到一张图片数据的前部。

2. 模型训练

定义和任务一中相同的网络结构，同时使用相同的超参数进行训练 10 轮（选择这个值是因为该模型训练 10 轮后仍未出现过拟合现象）。此时优化器改为 nn.SGD，因为如果使用 nn.Adam 将会导致梯度爆炸，使得后续训练时 loss 计算出 nan。

任务三

1. 数据预处理

在 FashionMNIST 上分别训练两个模型，数据预处理的方式如任务一和任务二。但是要注意的是，在标准化这一步，均值和方差要重新计算。

2. 模型训练

使用和任务一任务二中相同的超参数，皆训练 10 轮。

3. 实验结果与分析

任务一

训练 6 轮，选取初始学习率为 0.01，批大小设置为 64，绘制损失和预测准确率关于轮数的变化曲线，最终在测试集上获得了 97.51%准确率。

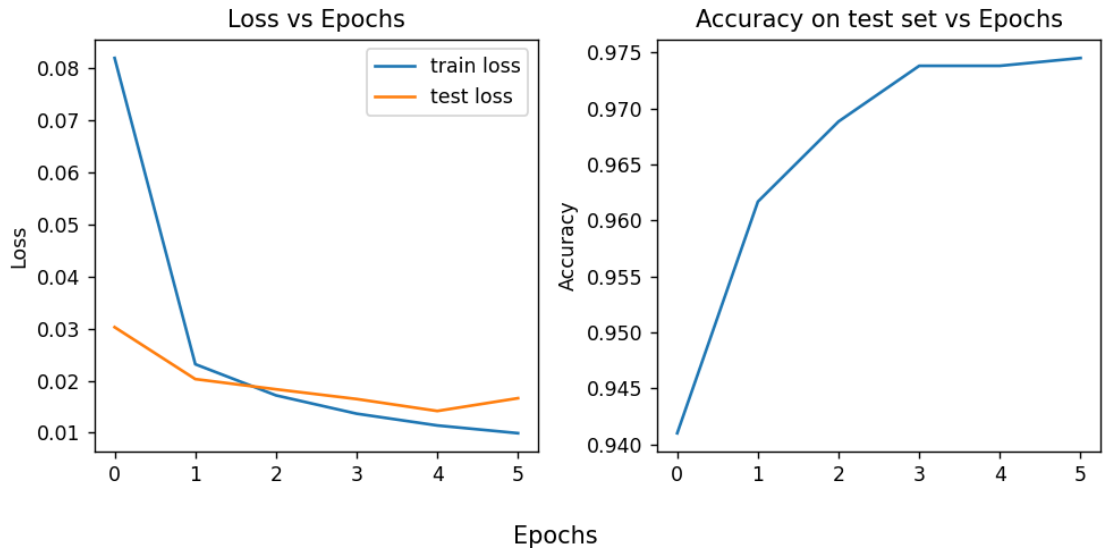

图 23 手动实现的神经网络的误差与在测试集上的准确率随轮数的变化曲线

任意打印一组结果，可以看到神经网络正确地完成了分类任务。

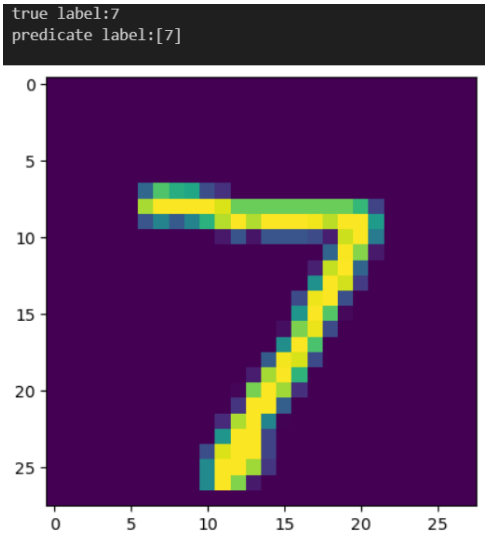

图 24 手动实现的神经网络的一组预测结果

任务二

训练 10 轮，选取初始学习率为 0.01，批大小设置为 64，绘制损失和预测准确率关于轮数的变化曲线，最终在测试集上获得了 95.92%准确率，略低于自己实现的网络。这有可能是默认参数的 SGD 表现不如 Adam，导致模型难以收敛到最优点。

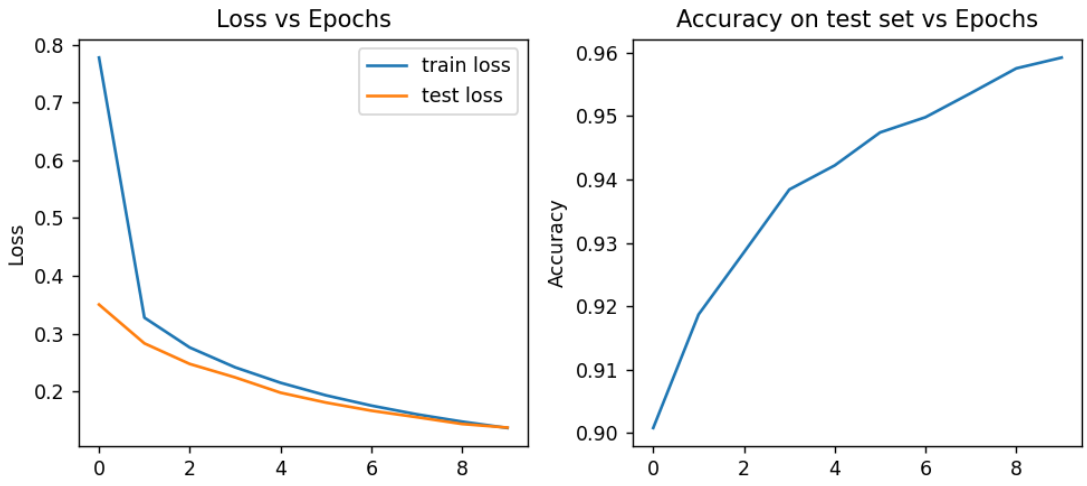

图 25 MindSpore 的神经网络的误差与在测试集上的准确率随轮数的变化曲线

任务三

用两个网络在 FashionMNIST 数据集上分别训练 10 轮，得到如下对比图。可以看到自己训练的网络无论是测试误差还是训练误差都要低于 MindSpore。而在准确率方面，自己的网络最终达到了 87.61%的分类效率，而 MindSpore 为 86.18%。

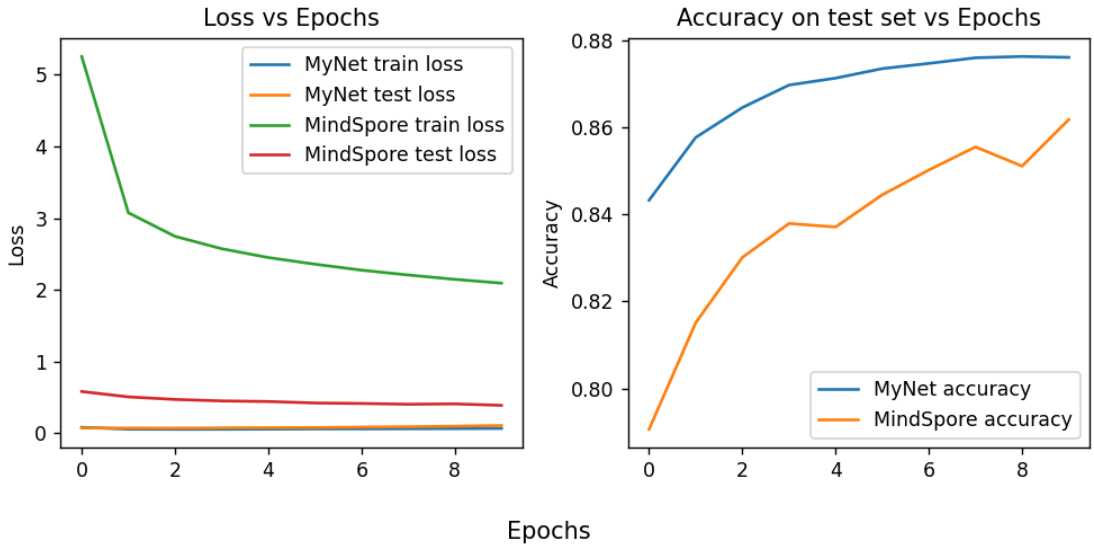

图 26 在 Fashion-MNIST 上两种网络的训练情况

4. MindSpore 学习使用心得体会

非常难用，如果不是课程要求我绝不会使用这个框架。生态非常差，遇到什么问题根本无从搜索，只能看文档，甚至是照搬示例，这导致学习成本大大增加。此外，许多异常都是 no decription，使得用户更难解决定位问题。它也没

有在 windows 上的 cuda 支持，训练速度十分受限。

心得体会

于 6.3 开始做，历时三天，终于在 6.6 的 23：37 完成了三个实验。有两天写到凌晨三四点，不由得感叹，我这么拼到底为了什么？我明明可以把代码写的简单，不必追求优雅、不必追求代码复用、不必追求接口统一，也没有必要去额外去研究什么 Adam、kd 树，毕竟以后日常工作中我也不可能会碰到需要具体实现这些无关代码的场景。写那么多报告，抄录那么多公式，反而耽误了我宝贵的复习其他科目的时间，要知道，这个学期我有 9 门课要考，大部分还是硬课，光这周末，我就要面临三门我几乎还没有复习过的课程。我完全可以只做好要求之内的任务，就拿到应有的分数。没有好处的事情，何必去做呢？

我不是一具傀儡，我有我的情绪、我的爱好、我对代码的追求。诚然，放弃掉那些，唯利益至上论，相比可以让人生过得更舒坦、更直接，然而在那条“光明大道”上，想必我已变成行尸走肉了吧。