实验二 朴素贝叶斯分类实验

1. 问题描述

- 概述
  - 利用朴素贝叶斯算法，对 MNIST 数据集中的测试集进行分类
- 数据说明
  - 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。
- 实验内容
  - 在课程学习中同学们已经学习了贝叶斯分类理论并掌握了其基本原理，即利用贝叶斯公式$p(w_j|x) = \frac{p(x|w_j)p(w_j)}{p(x)}$,对$p(w_j|x)$作出预测，由于$p(x)$为一固定值，所以一般不在计算过程中求得$p(x)$的具体值。在实际运用中，为了方便计算，通常假设数据特征之间相互独立，即

    $$p(x|w_j) = p(x_1|w_j) \cdot p(x_2|w_j) \ldots \cdot p(x|w_j), x \in \mathbb{R}^d$$

    这便是著名的朴素贝叶斯算法。
  - MNIST 数据集本身以二进制形式保存，所以首先需要选择合适的编程语言编写读写二进制数据的程序完成对图片、标记信息的初步提取工作。读取了图片信息后，发现每个像素点的值在[0,1]区间中，这是图像压缩后的结果，所以可以先将像素值乘以 255 再取整，得到每一个点的灰度值。将图像二值化，得到可以用于分类的 28×28 个特征向量以及对应的标签数据，之后便可以交由贝叶斯分类器进行学习。
  - 基于 MindSpore 平台实现算法，对相同的数据集进行训练，并与不使用 MindSpore 平台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。

2. 实现步骤与流程

1 数据预处理

1.1 数据读取

MNIST 数据集本身以二进制形式保存，所以首先需要选择合适的编程语言编写读写二进制数据的程序完成对图片、标记信息的初步提取工作。

1.2 数据增强

数据增强是通过一定的方式改变输入数据，以生成更多的训练样本，从而提高模型的泛化能力和效果。数据增强可以减少模型对某些特征的过度依赖，从而避免过拟合。在深度学习中，要求样本数量充足，样本数量越多，训练出来的模型效果越好，模型的泛化能力越强。但是实际中，样本数量不足或者样本质量不够好，这时就需要对样本做数据增强，来提高样本质量。

1.3 数据二值化

数据二值化是将图像中的每个像素值转换为 0 或 1 的过程。通常，二值化是通过将像素值与某个阈值进行比较来实现的。如果像素值大于阈值，则该像素被设置为 1；否则，设置为 0。这种处理方法可以显著减少图像数据的复杂性，便于后续的图像处理操作

2 计算每个类别的先验概率

首先，我们通过“计数”的方式计算出类别的先验概率：

$$p(w_i) = \frac{w_i\text{的样本数}}{\textit{总样本数}}$$

```
# 计算每个类别的先验概率
self.class_priors = np.zeros(n_classes)
for idx, c in enumerate(self.classes):
    self.class_priors[idx] = np.sum(y == c) / n_samples
```

3 计算每个特征在每个类别中的类条件概率

以同样的方式计算每个特征在每个类别中的类条件概率：

$$p(x|w_i) = \frac{\text{具有某个特征 x 在类别 i 的样本数}}{\text{该类别总样本数}}$$

```
# 计算每个特征在每个类别中的条件概率
self.feature_probs = np.zeros((n_classes, n_features, 2))
for idx, c in enumerate(self.classes):
    X_c = X[y == c]
    self.feature_probs[idx, :, 1] = (np.sum(X_c, axis=0) + 1) / (X_c.shape[0] + 2)  # 平滑
    self.feature_probs[idx, :, 0] = 1 - self.feature_probs[idx, :, 1]
```

4 估计待测样本的类别

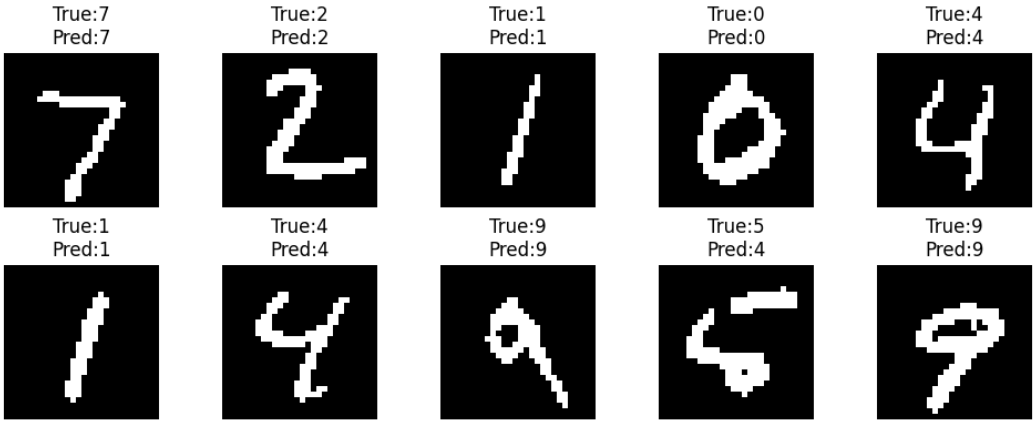
通过先前计算的先验概率和类条件概率以获得样本属于每一个类的后验概率，然后找出对应最大概率的标签，即可得到待测样本的预测类别。

3. 实验结果与分析

- 实验结果：

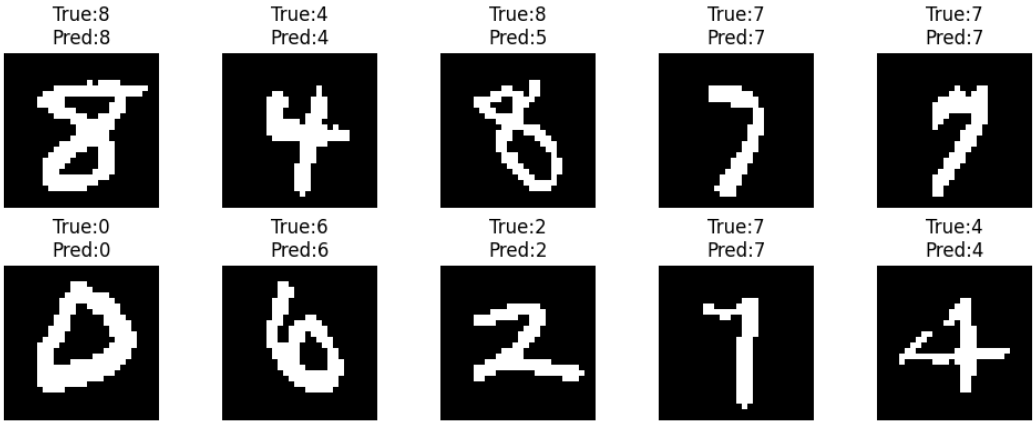
① 任务一：

- 模型: 手动实现的贝叶斯分类器
- 数据集: MNIST
- 测试集准确率: 84.33%
- 对部分在测试集上的预测结果进行打印，可以看到神经网络模型基本能够正确的完成分类任务



② 任务二：

- 模型: Sklearn 中的 BernoulliNB 贝叶斯分类器
- 数据集: MNIST
- 测试集准确率: 84.33%。
- 对部分在测试集上的预测结果进行打印，可以看到神经网络模型基本能够正确的完成分类任务。



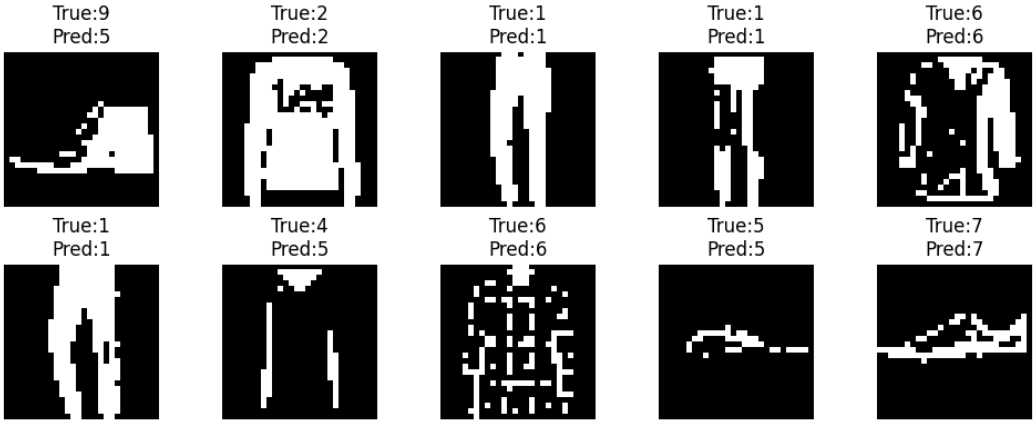
③ （加分项）任务三：

使用 MindSpore、sklearn 等平台提供的相似任务数据集（FashionMNIST）测试自己独立实现的算法，并与 MindSpore、sklearn 等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

- 数据集: FashionMNIST

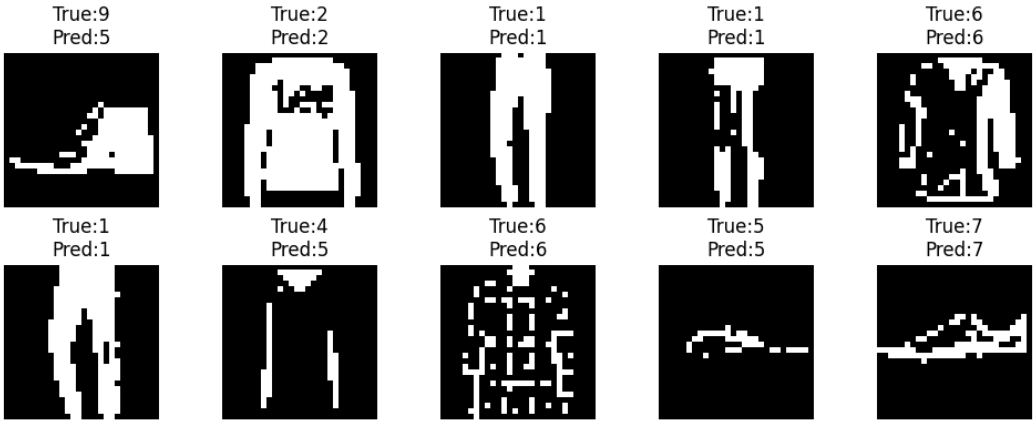
（1）子任务一

- 模型: 手动实现的贝叶斯分类器
- 测试集准确率: 64.56%
- 对部分在测试集上的预测结果进行打印，可以看到神经网络模型基本能够正确的完成分类任务



（2）子任务二

- 模型: Sklearn 中的 BernoulliNB 贝叶斯分类器
- 测试集准确率: 64.56%
- 对部分在测试集上的预测结果进行打印，可以看到神经网络模型基本能够正确的完成分类任务。



另外，各个实验完整的测试集分类结果分别保存在 task1_mnist_manual_test_predictions.csv, task2_mnist_sklearn_test_predictions.csv, task3_fashion_mnist_manual_test_predictions.csv, task3_fashion_mnist_sklearn_test_predictions.csv 中

■ **实验结果分析：**

1. MNIST 数据集上的实验（任务一 vs 任务二）

■ 控制变量: 数据集、模型类型（贝叶斯分类器）、训练和测试流程

■ 变化变量: 实现方式（手动实现 vs Sklearn 实现）

■ 实现方式对结果的影响

手动实现的贝叶斯分类器和 Sklearn 中的 BernoulliNB 贝叶斯分类器在 MNIST 数据集上的测试集准确率均为 84.33%。

这表明，尽管实现方式不同，但在相同的数据集和相同的模型类型下，两种实现

方式的效果是一致的。说明手动实现的贝叶斯分类器在功能上达到了 Sklearn 实现的效果。

2. FashionMNIST 数据集上的实验（任务三）

- 控制变量: 数据集、模型类型（贝叶斯分类器）、训练和测试流程
- 变化变量: 实现方式（手动实现 vs Sklearn 实现）
- 实现方式对结果的影响

手动实现的贝叶斯分类器和 Sklearn 中的 BernoulliNB 贝叶斯分类器在 FashionMNIST 数据集上的测试集准确率均为 64.56%。

这再次表明，尽管实现方式不同，但在相同的数据集和相同的模型类型下，两种实现方式的效果是一致的。

3. 数据集对模型性能的影响（MNIST vs FashionMNIST）

- 控制变量: 模型类型（贝叶斯分类器）、实现方式（手动实现 vs Sklearn 实现）、训练和测试流程
- 变化变量: 数据集
- 数据集复杂度对结果的影响

在 MNIST 数据集上，手动实现和 Sklearn 实现的贝叶斯分类器均达到了 84.33%的准确率。

在 FashionMNIST 数据集上，手动实现和 Sklearn 实现的贝叶斯分类器均达到了 64.56%的准确率。

MNIST 数据集是手写数字数据集，而 FashionMNIST 数据集是服饰图像数据集。相比 MNIST 数据集，FashionMNIST 数据集更复杂，类别间的区分难度更大。因此，在相同模型和相同实现方式下，模型在 FashionMNIST 数据集上的准确率显著低于 MNIST 数据集。

4. MindSpore 学习使用心得体会

朴素贝叶斯分类器以其简洁的模型结构和相对较低的计算复杂度，在多个应用场景中展现出了不俗的性能。其核心优势在于能够直接从数据中学习出所需的数据特征，作为一种生成模型，它为我们提供了一种直观的方式来理解数据的内在结构。然而，这种方法的局限性也显而易见：当数据不满足独立同分布（IID）的假设时，其分类效果可能会大打折扣。

在本次实验中，我使用了 FashionMNIST 和 MNIST 这两个经典的数据集，它们的数据特性与朴素贝叶斯分类器的假设高度契合，因此实验结果十分理想。通过实践，我深刻体会到，在应用任何机器学习算法之前，对数据特性的深入理解和分析是至关重要的。

此外，这次实验也加强了我们对于 numpy 和 pandas 等数据处理工具包的熟练程度。这些工具包在处理大量数据时提供了强大的支持，帮助我们更加高效地进行数据预处理和特征工程。通过实践，我们进一步提高了自己的编程能力，并加深了对数据集复杂性对于模型训练效果的影响的理解。

5. 代码附录

限于实验报告篇幅原因，未在实验报告中粘贴完整代码，代码详见代码附件