机器学习引论期末作业

摘要

本实验基于MNIST手写数字图像数据集，设计并实现了一种高效的图像分类模型。为充分利用全部70000个样本并避免维度灾难问题，先采用PCA进行降维，分别测试10、20、30与50维的分类效果。在分类算法选择上，使用K近邻（KNN）与支持向量机（SVM）模型，结合交叉验证优化超参数配置。实验显示，KNN在30维、k=5时达到97.27%准确率，SVM在50维、C=10时准确率高达98.33%，均具有出色的分类性能。整个训练过程借助cuML与sklearn库，运行于WSL2环境中的Windows 11平台，有效利用GPU并行计算资源，实现了在大样本条件下的快速训练。实验流程包括数据预处理、降维、模型训练、调参及性能评估，结果表明降维+经典分类器方法在中等规模图像分类任务中兼具效率与精度，具有良好应用前景。

实验代码请见：Frank-LuHao/Machine_Learning_2025_Spring (github.com)

一、任务描述

基于手写数字图像数据集 MNIST ，设计并实现一种有效的图像分类模型。通过本学期课程所介绍的机器学习算法，训练一个分类器，使其在测试集上达到**90%以上的分类准确率**。此外，还需报告模型在该任务中的**F1 值（F-measure）**，以综合评估分类性能在精确率（Precision）与召回率（Recall）之间的平衡。

具体要求包括：

- 明确说明所采用的模型架构、特征处理方法及训练策略，并阐释各设计选择的合理性；
- 通过多次实验统计结果的平均值（mean）与标准差（standard deviation），反映模型性能的稳定性；
- 给出调参过程及最终使用的关键超参数；
- 报告实验所用硬件环境（如 GPU 型号、内存等）及训练与推理阶段所花费的时间。

二、方法选择

由于MNIST数据集中图像有28 * 28 = 784个特征维度，可能会存在维灾 (Curse of Dimensionality) 问题，进而导致：

- 计算量巨大，特别是在MNIST中总共存在70000个有效数据的情况下
- 数据稀疏，在高维空间中数据分布极其稀疏，少量的关键特性分布在所有维度张成的空间的概率接近0
- 距离度量失效，在高维空间中数据的真实分布和关系被欧氏距离的均等加权所掩盖
- 过拟合风险，高维空间中模型更容易扑捉到噪声与异常值

因此，为了有效利用scaling law，发挥70000个数据量的优势，先对数据集进行降维处理，然后使用分类算法进行训练与评估。

降维算法选择：

降维算法	分析
PCA	去除冗余信息，计算效率高，简单快速，能够处理新样本
CCA	弱监督，多子空间，一般用于多模态
LDA	有监督，多子空间，能够有效利用标签数据进行降维
LLE/LE	学习局部欧式空间，但是计算效率低，难以处理新样本
NPE/LPP	学习局部欧式空间，计算效率比LLE/LE好，能够处理新样本

综合以上分析，以及小样本上的快速验证结果

- 优先选择PCA
- LLE/LE效果很好，但是大样本计算慢
- 在`cuML`库中，其它算法现有接口不完善

分类算法选择：

选择KNN与SVM（由于Perceptron同样是学习一个超平面做线性分类，但是没有SVM的最大间隔，因此不考虑）

三、实验流程

为了有效利用GPU的并行计算能力，使用`cuML`库与`sklearn`库进行实验

实验环境：

- Window11 + WSL2 + VS Code 1.101.0
- `cuML` 23.04 + `sklearn` 1.6.1
- intel(R) i7-13700H + RTX 3050 4GB

实验步骤：

①数据加载 & 预处理

加载70000个数据，并将原有训练集与测试集合并，以便后续降维

```
transform = transforms.Compose([
    transforms.ToTensor(),
])

mnist_train = datasets.MNIST(root='./data', train=True, download=True, transform=transform)
mnist_test = datasets.MNIST(root='./data', train=False, download=True, transform=transform)

# 合并训练集和测试集
X = np.concatenate([
    mnist_train.data.reshape(len(mnist_train), -1).numpy(),
    mnist_test.data.reshape(len(mnist_test), -1).numpy()
], axis=0).astype(np.float32)
y = np.concatenate([
    mnist_train.targets.numpy(),
    mnist_test.targets.numpy()
], axis=0)
```

对输入特征归一化

```
# 标准化
scaler = StandardScaler()
X_pca = scaler.fit_transform(X_pca)
```

②数据降维

- 使用PCA降维并保存结果至本地
- 为了方便后续超参数选择，分别降维至10、20、30、50维

平均降维耗时：0.33s

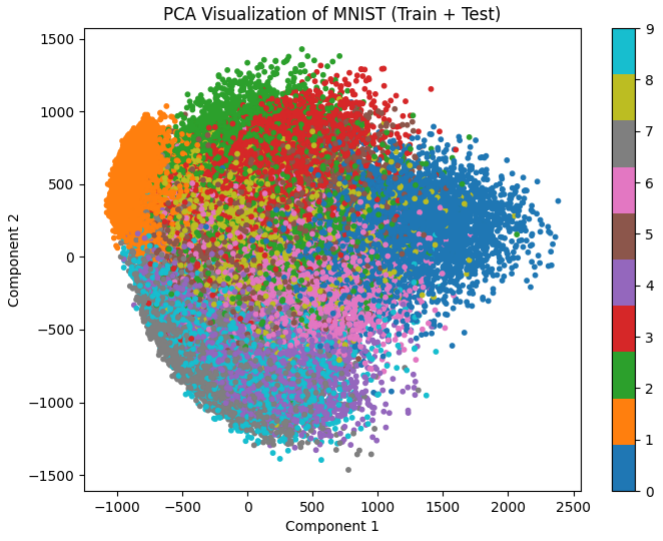
```
=== PCA降维，维度：10 ===
降维后结果已保存。
降维后形状：  (70000, 10)
PCA降维耗时：0.33 秒

=== PCA降维，维度：20 ===
降维后结果已保存。
降维后形状：  (70000, 20)
PCA降维耗时：0.32 秒

=== PCA降维，维度：30 ===
降维后结果已保存。
降维后形状：  (70000, 30)
PCA降维耗时：0.33 秒

=== PCA降维，维度：50 ===
降维后结果已保存。
降维后形状：  (70000, 50)
PCA降维耗时：0.34 秒
```

降维后前两维可视化效果：



③模型训练

- 加载降维数据，将数据划分为训练集、验证集与测试集
- 在训练集上分别对 KNN 与 SVM 进行训练

平均训练耗时 (s) [1]：

模型/维度	10	20	30	50
SVM	6.20	5.96	33.89	60.26

④超参数选择

使用 k-fold cross-validation，在训练集上训练，在验证集上测试，通过测试结果选择超参数

(1) 降维维度选择 (准确率%) [2]：

模型/维度	10	20	30	50
KNN	92.51	96.57	97.03	96.43
SVM	93.43	97.27	97.96	98.07

分析结果可知：

- 对于KNN，当维度过高时，维灾会导致准确度下降，因此选择30维数据
- 对于SVM，维度越高，准确率越高，但是训练时间也随之增加，选择50维数据

(2) KNN中 k 的选择 (准确率%) [3]：

模型/参数	1	3	5	7	9
KNN	96.80	97.00	97.03	96.93	96.78

分析结果可知，选择 k=5

(3) SVM中正则化参数选择 (准确率%) [4]：

模型/参数	0.1	1	10	100
SVM	96.11	98.07	98.31	98.31

分析结果可知，选择C=10.0

⑤模型评估

将模型在训练集上测试，报告实验结果如下：

(1) KNN

> *参数：dim = 30, k = 5, weight = uniform, metric = minkowski*

```
=== 在测试集上评估 KNN 模型 ===
测试集准确率: 0.9727
测试集F1分数（macro）: 0.9725
测试集分类报告:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1381
           1       0.97      0.99      0.98      1575
           2       0.98      0.96      0.97      1398
           3       0.97      0.97      0.97      1428
           4       0.99      0.96      0.97      1365
           5       0.97      0.96      0.97      1263
           6       0.97      0.99      0.98      1375
           7       0.96      0.98      0.97      1459
           8       0.98      0.94      0.96      1365
           9       0.96      0.97      0.96      1391

    accuracy                           0.97     14000
   macro avg       0.97      0.97      0.97     14000
weighted avg       0.97      0.97      0.97     14000
```

(2) SVM

> *参数：dim = 50, kernel=rbf, gamma = 1 / (n_features * X.var())*

```
=== 在测试集上评估 SVM 模型 ===
测试集准确率: 0.9834
测试集F1分数（macro）: 0.9833
测试集分类报告:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1381
           1       0.99      0.99      0.99      1575
           2       0.98      0.98      0.98      1398
           3       0.99      0.97      0.98      1428
           4       0.99      0.98      0.98      1365
           5       0.98      0.98      0.98      1263
           6       0.98      0.99      0.99      1375
           7       0.98      0.98      0.98      1459
           8       0.98      0.98      0.98      1365
           9       0.97      0.98      0.98      1391

    accuracy                           0.98     14000
   macro avg       0.98      0.98      0.98     14000
weighted avg       0.98      0.98      0.98     14000
```

四、结论

本实验通过对MNIST数据集的深入分析，结合PCA降维与KNN、SVM等经典分类算法，实现了在大样本条件下的高效图像分类任务。实验结果表明：合理的降维能够显著提升模型训练效率与泛化能力，KNN和SVM在精度与F1值方面均表现优异，最高准确率分别达到97.27%和98.33%。此外，通过系统的超参数调优与交叉验证，进一步保证了模型性能的稳定性与可靠性。整体实验验证了“降维+传统分类器”在中等规模图像分类中的实用性，为今后在资源受限或模型部署环境中实现高效学习提供了有益参考。

---

1. 由于KNN没有显示的训练过程，因此这里忽略 ↩

2. 这里控制参数 **KNN** k=5, weights=uniform, **SVM** C=1.0, kernel=rbf, 其它超参数保持默认 ↩

3. 这里控制参数 dim=30, weights=uniform ↩

4. 这里控制参数 dim =50, kernel=rbf, 其它超参数保持默认 ↩