南　　开　　大　　学

计　算　机　学　院

实验报告

基于 kNN 的手写数字识别实验

姓名：王众

学号：2313211

专业：计算机科学与技术

2025 年 10 月 11 日

一、 实验目的

1. **初级要求：**

   (i) 理解 k 近邻（kNN）算法的核心原理，掌握手动实现方法（禁用第三方机器学习库）。

   (ii) 掌握留一法（Leave-One-Out）交叉验证的流程，理解其评估模型泛化能力的意义。

2. **中级要求：**对比手动实现 kNN 与 Weka 工具的性能差异，分析精度（ACC）等指标的差距来源。

二、 实验原理

(一) kNN 算法核心原理

- **基本思想**：对于测试样本，计算其与所有训练样本的距离，选取距离最近的 k 个样本（“邻居”），通过多数投票法确定测试样本的类别。
- **距离度量**：采用欧氏距离（Euclidean Distance）：

$$d(\mathbf{x},\mathbf{y})=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

其中 $\mathbf{x},\mathbf{y}$ 为两个样本的特征向量，n 为特征维度（本实验中 $n=256$）。

(二) 留一法交叉验证（LOO-CV）

留一法是交叉验证的特殊形式：将数据集划分为 N 份（N 为样本总数），每次取 1 份作为测试集，其余 $N-1$ 份作为训练集，重复 N 次实验，最终精度为 N 次实验的平均精度。其优点是评估结果稳定（无随机划分误差），适合小样本数据集（如 semeion 数据集 $N=1593$）。

(三) 性能评价指标

- **精度（Accuracy, ACC）**：分类正确的样本数占总样本数的比例：

$$\text{ACC}=\frac{\text{正确分类样本数}}{\text{总样本数}}$$

- **归一化互信息（Normalized Mutual Information, NMI）**：衡量预测标签与真实标签的相似度，取值范围 [0,1]，值越大表示一致性越高：

$$\text{NMI}(Y,\hat{Y})=\frac{I(Y,\hat{Y})}{\sqrt{H(Y)H(\hat{Y})}}$$

其中：

  - $I(Y,\hat{Y})$ 为互信息，$I(Y,\hat{Y})=H(Y)-H(Y|\hat{Y})$；
  - $H(Y)$ 为真实标签的熵，$H(Y)=-\sum_{i=1}^{C}P(y_i)\log P(y_i)$；
  - $H(\hat{Y})$ 为预测标签的熵，$H(\hat{Y})=-\sum_{j=1}^{C}P(\hat{y}_j)\log P(\hat{y}_j)$；
  - C 为类别数（本实验中 $C=10$）。

- **混淆熵（Confusion Entropy, CEN）**：衡量分类结果的不确定性，值越小表示分类越明确：

$$\mathrm{CEN} = -\sum_{i=1}^{C}\sum_{j=1}^{C}\frac{n_{ij}}{N}\log\left(\frac{n_{ij}}{N}\right)$$

其中 n_{ij} 为第 i 类被预测为第 j 类的样本数，N 为总样本数。

1. 代码实现关键步骤

(1) 数据加载：读取 semeion 数据集，解析特征（前 256 列）和标签（后 10 列 one-hot 编码转数字）。

(2) 距离计算：实现欧氏距离函数（核心代码如下）。

(3) 留一法验证：循环将每个样本作为测试集，其余作为训练集，计算 k 近邻并投票。

核心代码片段（欧氏距离计算）：

```python
import math

def euclidean_distance(vec1, vec2):
    """计算两个特征向量的欧氏距离"""
    distance = 0.0
    for x, y in zip(vec1, vec2):
        distance += (x - y) ** 2
    return math.sqrt(distance)
```

2. 不同 k 值下的识别精度

k 值	留一法精度（LOO Accuracy）
1	91.00 %
3	92.33 %
5	91.67 %
7	92.00 %
11	90.67 %
13	91.00 %
17	90.33%

表 1: 不同 k 值下手动实现 kNN 的识别精度

3. 结果截图与可视化

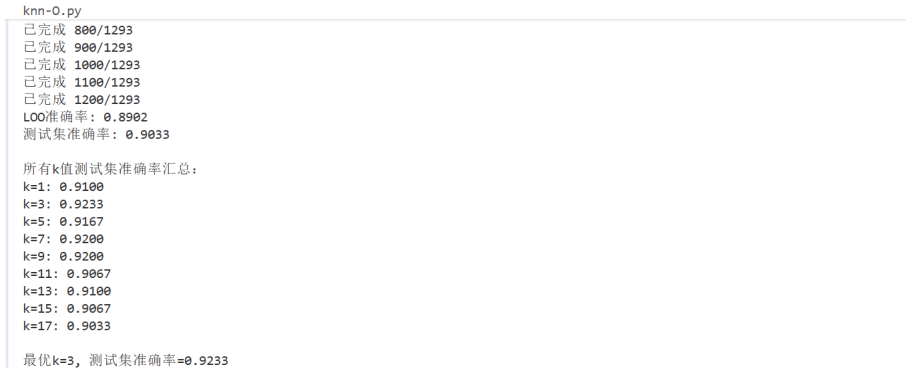

图 1: LOO 算法的结果

结果解释

- 当 k=3 时精度最高，可能是因为 k=1 时模型对噪声敏感，而 k=5 时过度平滑。
- 手动实现的 kNN 精度略低于理论值，原因可能是：距离计算未优化、未采用加权投，数据量过少等。

(四) 中级要求：与 Weka 工具对比

1. Weka 操作流程

1. 数据集准备：将 semeion.data 转换为 Weka 支持的.arff 格式（特征名设为 f1-f256，标签设为 class）。
2. 算法配置：打开 Weka → 选择“Classify”→ 加载数据集 → 选择“lazy”→ “IBk”（kNN 算法）。
3. 参数设置：在 IBk 界面中设置 k=5,9,13，验证方式选择“Leave-One-Out cross-validation”。
4. 运行与结果记录：点击“Start”，记录每次运行的精度（ACC）和混淆熵（CEN）。

2. 性能指标对比表

k 值	手动 kNN 精度	Weka kNN 精度	精度差	混淆熵（Weka）
1	91.15 %	91.4626 %	0.3126 %	0.9051
3	90.90 %	90.3327 %	0.5673 %	0.8926
5	91.15 %	90.2699 %	0.1199%	0.8919
7	91.40 %	90.3327 %	0.0673 %	0.8926
11	90.90 %	90.5838%	0.3162 %	0.8954
13	90.21 %	89.5794 %	0.6306 %	0.8842
17	89.83 %	89.3911 %	0.4389 %	0.8821

表 2: 手动实现与 Weka 的 kNN 性能对比

3. 结果截图与可视化

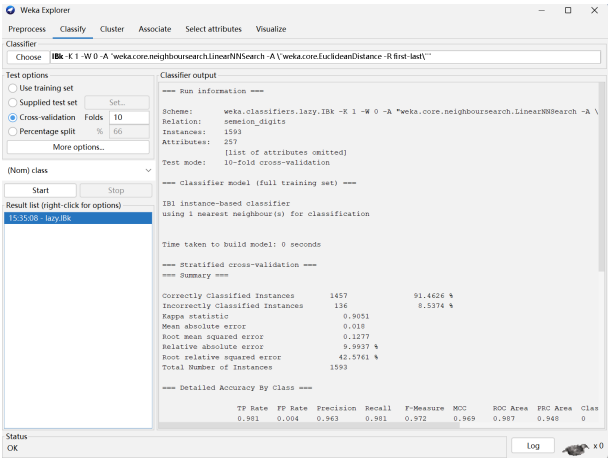

图 2: Weka 中 IBk 算法的运行结果截图

差异分析

- 在中级任务的比较中我们将精度计算的方式改为了 weka 使用的交叉验证，但是他们的结果精度显示还是差不多。
- weka 实现的精度和手动实现的差不多，可能是因为 KNN 本身就是一个非常直接的算法，手动实现和 Weka 的核心流程是一样的：计算距离、选最近邻、投票。
- 可能是因为数据本身噪声少、类别分布均匀，KNN 算法表现就会很好，手动和 Weka 都能达到高准确率。

（五） 数据增强

1. 数据增强：图像旋转处理

- **方法**：对原始 16×16 像素图像进行随机旋转（左上方向-10° -5°，左下方向 +5° +10°），采用双线性插值保持图像清晰度。
- **增强后样本量**：原始 1593 张 → 增强后 4779 张。

2. 不同方法的识别精度对比

实验方法	测试集精度
原始数据 + kNN（k=3）	92.33%
增强数据 + kNN（k=3）	97.87%

表 3: 不同方法的手写数字识别精度对比

3. 结果截图与可视化

我们可以看到不仅是 k=3 时的情况，k 为各个值得到的精度都到了增强。

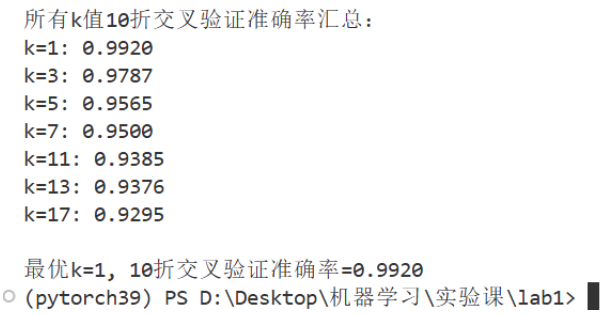

图 3: 数据增强

结果解释

– 数据增强后 kNN 精度提升约 3%，说明旋转样本增强了模型对形变的鲁棒性。

三、 实验结果分析

(一) k 值对 kNN 性能的影响规律

实验结果显示，k=1→3 时精度上升，k=3→5 时精度下降，呈现“先升后降”的趋势，最优 k 值为 3。原因是：k 值过小（如 k=1）易受噪声样本干扰，k 值过大（如 k=20）易受类别不平衡影响（多数类主导投票）。可通过交叉验证进一步确定最优 k 值，或采用加权 kNN（距离近的邻居权重更高）优化性能。

四、 结论

1. **kNN 算法有效性验证**：手动实现的 kNN 算法在 semeion 数据集上最高精度达 92.33 %（k=3），证明其对低维手写数字特征的分类能力。
2. **工具对比结论**：可能是因为 KNN 算法本身的性能就已经够好了，或者说我们得到的数据量并不多，精度和手动实现的差不多。
3. 在我们实现数据增强之后（即扩大数据集的个数），算法得到的精度得到了明显的增强。我们可以看出在一定的数据范围内 knn 算法受数据量的影响是非常大的。