南開大學
Nankai University

计算机学院

机器学习实验报告

基于 kNN 的手写数字识别实验

姓名：周重天

学号：2311082

专业：计算机科学与技术

2025 年 10 月 11 日

目录

```
● (base) ➜  work1 python -u "/Users/zhouzhongtian/vscode/machinelearning/work1/main.py"
  k=1 训练集 LOO 准确率 = 0.9118
  k=2 训练集 LOO 准确率 = 0.8995
  k=3 训练集 LOO 准确率 = 0.9103
  k=5 训练集 LOO 准确率 = 0.9157
  k=7 训练集 LOO 准确率 = 0.9126
  k=11 训练集 LOO 准确率 = 0.8987
  k=13 训练集 LOO 准确率 = 0.8971
  k=17 训练集 LOO 准确率 = 0.8848
  k=19 训练集 LOO 准确率 = 0.8786
  最佳 k 值: 5
  测试集准确率: 0.9100
```

1 实验名称

基于 kNN 的手写数字识别实验

2 实验目的

1. 实现 k 近邻（k-Nearest Neighbor, kNN）分类器
2. 掌握留一法（Leave-One-Out, LOO）交叉验证流程
3. 对比自实现结果与 Weka 内置 kNN 精度
4. 学会用图表展示实验过程与结论

3 实验结果

将 k 的取值范围扩大到 20 以内的质数以及 1 内，经过几次运行，结果均为 k=7 和 5 时在训练集上取得最高 loo 准确率，以下为其中一个结果 下表为多次实验所得数据的汇总：

表 1: kNN 算法实验结果汇总

实验次数	最佳 k 值	训练集 LOO 准确率	测试集准确率
1	5	0.9149	0.9133
2	5	0.9149	0.9067
3	5	0.9126	0.9033
4	1	0.9118	0.9100
5	5	0.9126	0.9133
6	5	0.9157	0.9100
平均值	–	**0.9138**	**0.9094**

4 关键代码

以下部分对本次实验的一些关键代码展开解释：

4.1 数据预处理

Listing 1: 数据加载与预处理

```
# 加载训练集
train_raw = np.loadtxt('semeion_train.txt')
train_X, train_y = train_raw[:, :256], np.argmax(train_raw[:, 256:], 1)

# 加载测试集
test_raw = np.loadtxt('semeion_test.txt')
test_X, test_y = test_raw[:, :256], np.argmax(test_raw[:, 256:], 1)
```

这段代码实现了独立训练集和测试集的加载和预处理:分别从 `semeion_train.txt` 和 `semeion_test.txt` 加载数据，`raw[:, :256]` 提取前 256 列作为特征矩阵 X，`np.argmax(raw[:, 256:], 1)` 将独热编码的标签转换为类别标签 y，其中 `axis=1` 表示沿行方向寻找最大值索引（也就是找到独热编码中的 1）。

4.2 距离计算函数

Listing 2: 欧氏距离计算

```
def euclidean_distance(test_sample, train_X):
    distances = np.sqrt(np.sum((train_X - test_sample) ** 2, axis=1))
    return distances
```

该函数使用 numpy 库实现了欧氏距离的计算：这里通过 `(train_X - test_sample) ** 2` 计算各维度差值的平方，然后 `axis=1`：沿特征维度求和，得到每个训练样本与测试样本的距离，实现了根据欧氏距离公式：$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

4.3 LOO 交叉验证核心算法

Listing 3: Leave-One-Out 交叉验证

```
for i in range(n_samples):
    test_sample = X[i:i+1]
    test_label = y[i]

    train_X = np.concatenate([X[:i], X[i+1:]], axis=0)
    train_y = np.concatenate([y[:i], y[i+1:]], axis=0)

    distances = euclidean_distance(test_sample, train_X)
    k_nearest_indices = np.argsort(distances)[:k]
    k_nearest_labels = train_y[k_nearest_indices]
```

这里根据 LOO 交叉验证的方法,我们每次从训练集中取出一个样本 i 作为测试集(`test_sample = X[i:i+1]`)，并将其余样本作为训练集 (`train_X = np.concatenate([X[:i], X[i+1:]], axis=0)`)，计算测试集样本 i 与其余样本的距离并找到距离最近的 k 个样本的标签。
特别注意的是，这里一共进行了 n_sample 次循环，其中 n_sample 是 X 中的行数，表示针对每一个 X 的样本，我们都将其作为目标分类样本训练了一次，确保最大化利用训练集。

4.4 投票机制

Listing 4: 多数投票与平局处理

```
unique_labels, counts = np.unique(k_nearest_labels, return_counts=True)
max_count = np.max(counts)
max_labels = unique_labels[counts == max_count]

if len(max_labels) > 1:
    predicted_label = np.random.choice(max_labels)
else:
    predicted_label = max_labels[0]
```

我们在这里获取到了所有的标签类别以及对应在 k 个最近邻点中出现的次数（这两个变量数组是一一对应的），然后找到最大的出现次数 count，并针对存储的所有标签类别的数组 `max_labels` 通过 `counts == max_count` 这个筛选条件进行选择，当筛选结果大于一的时候说明有平票的情况，此时使用随机数进行随机选择。

4.5 模型选择与评估流程

Listing 5: 最佳 k 值选择

```
k_values = generate_primes(20)
k_values = [1] + k_values
best_k = None
best_accuracy = 0
for k in k_values:
    acc = loo_eval(train_X, train_y, k)
    if acc > best_accuracy:
        best_accuracy = acc
        best_k = k

final_accuracy = knn_eval(train_X, train_y, test_X, test_y, best_k)
```

我们在这里对生成的小于 20 的所有质数 k 加上 1 组成候选 k 值集合，通过在训练集上进行 LOO 交叉验证来选择最佳 k 值，然后选择出最终结果正确率最高的那个 k 作为训练结果，最后在独立的测试集上对这个最佳 k 进行评估，获得无偏的性能估计。

5 weka 对比

特别的，我们下载了实验指导书中的 weka 工具，并使用其内置的 knn、loo 进行了测试并与我们自己写的代码进行了对比，对比数据如下：

表 2: 自写算法与 Weka 的 LOO 精度对比

k	自写 LOO 精度	Weka LOO 精度	差异（绝对值）
1	91.59%	91.651%	0.061%
3	90.65%	90.2072%	0.443%
5	90.52%	90.7094%	0.189%

对比如上图，并没有出现指导书中所说的差异精度差异绝对值大于 0.5% 的情况，但是我查阅了一些资料，发现 weka 中的 ibk 算法的选择方案使用是，平票时固定选择第一个分类，而我的代码实现的是平票时随机选择一个分类，或许，在 k 值变大的情况下，会因此出现精度差异绝对值大于 0.5% 的情况。因此，我试着对比 k = 7 的情况 确实，调大 k 值之后，出现了差异绝对值变大的情况。另外，

表 3: 自写算法与 Weka 的 LOO 精度对比

k	自写 LOO 精度	Weka LOO 精度	差异（绝对值）
7	0.9102%	0.8945%	1.57%

再观察 weka 的输出信息，我发现他与我的自写 LOO 代码一样使用的是欧氏距离，因此这一差异绝对值变大的情况并非源于距离定义的影响。

6 总结

本次实验中，我对 KNN 分类算法有了一定的了解，熟悉了其执行的过程，并使用 python 对其进行了实现。在此过程中，我同样对 loo 的概念有了初步的认识。我平时学习中使用 python 的频率不高，在这次实验中，我对 python 的 numpy 库的语法、python 的列表切片的语法有了一定的熟悉。此外，还在人工智能导论课程之外，一定程度学习了训练集和测试集的使用。在实验时，我一开始对训练集和测试集的用途产生了混淆，还一度拥有“这 knn 有什么可训练的地方，不就是直接跑过程然后看分类结果吗”的疑问，在与同学交流之后，我才真正了解，通过训练集我们筛选出了最佳的 k 值，测试集是用来测试这个最佳的 k 值的表现的。

7 GitHub

本次实验源码已经上传到 GitHub 仓库中，GitHub 链接为：https://github.com/sskystack/machinelearning.git，如有错漏欢迎指出。