模式识别与机器学习课程设计报告

学号：

姓名：

班级：

2022年12月

关于手写数字识别的特征工程

**摘要**

手写数字识这项技术是光学字符识别(Optical Character Recognition,简称OCR)的一个重要分支。手写数字识别的基本原理是将输入样本数字与对应的标准样本数字进行模式匹配，将具有最大类似度的样本数字作为识别结果。在整个识别过程中，关键的是对样本特征提取方法的选择和分类器的设计，快速而全面的特征提取方法和高效而准确的分类器设计，决定着识别系统的分类效果和性能优劣。数字识别有广阔的应用前景和实用价值，并且在文献检索、办公自动化、邮政自动分拣、银行票据自动识别 等领域也有重要应用，因此手写数字识别有着重大的现实意义。

关于手写数字识别的特征工程已经在很多行业领域发挥了极大的作用，也逐渐进入人们的生活。对手写数字识别的特征进行降维，首先画出累计方差贡献率曲线，找出最佳降维后维度的范围，再画出降维后维度的学习曲线，继续缩小最佳维度的范围并细化学习曲线，找出降维后的最佳维度：16维特征。为查看模型效果，使用默认的KNN参数和随机森林进行交叉验证。对比使用主成分分析（PCA）、局部线性嵌入（LLE）和 T-随机邻近嵌入（T-SNE）三种模型进行降维后的数据维数、所花时间以及交叉验证后的准确率，局部线性嵌入（LLE）模型优于T-随机邻近嵌入（T-SNE）0.0017，但T-随机邻近嵌入（T-SNE）的维度受限，只能是3维特征，主成分分析（PCA）(16维)和局部线性嵌入（LLE）(18维)会比较适合一点，数据集要保留的数据量更大， 其中局部线性嵌入（LLE）为了调成最优时间成本很高，花了2005秒，综合来看PCA会在准确率，数据量以及时间成本上更合适。

关键词：**手写数字识别、特征降维、交叉验证**

**1 前言**

### 1.1 研究背景与意义

手写数字识这项技术是光学字符识别(Optical Character Recognition,简称OCR)的一个重要分支，主要分为脱机手写数字识别和联机手写数字识别。手写数字识别的基本原理是将输入样本数字与对应的标准样本数字进行模式匹配，将具有最大类似度的样本数字作为识别结果。在整个识别过程中，关键的是对样本特征提取方法的选择和分类器的设计，快速而全面的特征提取方法和高效而准确的分类器设计，决定着识别系统的分类效果和性能优劣。

由于手写数字识别没有上下文，不存在语义相关性，而一些部门对数字的要求又相对较高，如银行报表、邮政编码、财务报表等，所以对其识别精度也需要达到更高的水平。因为数字识别有广阔 的应用前景和实用价值，并且在文献检索、办公自动化、邮政自动分拣、银行票据自动识别 等领域也有重要应用，因此国内外许多研究学者也在为进一步提高识别率和通用性而努力. 并且，现今越来越多的智能信息处理系统和终端配备了电子摄像或扫描设备，数字识别可广 泛地应用于这类设备的输入、电子商务系统的签名鉴别等领域中，为文字信息处理的自动化 及进一步提升计算机智能输入提供重要的理论意义及广阔的应用前景,因此手写数字识别有着重大的现实意义。

### 1.2 研究内容

对手写数字识别的特征进行降维，读取并查看手写数字的数据，画出累计方差贡献率曲线，找出最佳降维后维度的范围，再画出降维后维度的学习曲线，继续缩小最佳维度的范围并细化学习曲线，找出降维后的最佳维度，查看模型效果。使用默认的KNN参数和随机森林进行交叉验证；选择效果好的进行机器学习后的交叉验证，对比使用主成分分析（PCA）、局部线性嵌入（LLE）和 T-随机邻近嵌入（T-SNE）三种模型进行降维后的数据维数、所花时间以及交叉验证后的准确率，选出最合适的模型。

# 2 项目原理

### 2.1 主成分分析（PCA）

主成分分析方法，是一种使用最广泛的数据降维算法。PCA的主要思想是将n维特征映射到k维上，这k维是全新的正交特征也被称为主成分，是在原有n维特征的基础上重新构造出来的k维特征。

PCA的工作就是从原始的空间中顺序地找一组相互正交的坐标轴，新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第1,2个轴正交的平面中方差最大的。依次类推，可以得到n个这样的坐标轴。通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面k个坐标轴中，后面的坐标轴所含的方差几乎为0。

于是，我们可以忽略余下的坐标轴，只保留前面k个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为0的特征维度，实现对数据特征的降维处理。

实现步骤：

1.去除平均值

2.计算协方差矩阵

3.计算协方差矩阵的特征值和特征向量

4.将特征值排序

5.保留前N个最大的特征值对应的特征向量

6.将原始特征转换到上面得到的N个特征向量构建的新空间中（最后两步，实现 了特征压缩）

### 2.2 局部线性嵌入（LLE）

LLE属于流形学习的一种。因此我们首先看看什么是流形学习。流形学习是一大类基于流形的框架。数学意义上的流形比较抽象，不过我们可以认为LLE中的流形是一个不闭合的曲面。这个流形曲面有数据分布比较均匀，且比较稠密的特征，有点像流水的味道。基于流行的降维算法就是将流形从高维到低维的降维过程，在降维的过程中我们希望流形在高维的一些特征可以得到保留。

一个形象的流形降维过程如下图。我们有一块卷起来的布，我们希望将其展开到一个二维平面，我们希望展开后的布能够在局部保持布结构的特征，其实也就是将其展开的过程，就想两个人将其拉开一样。

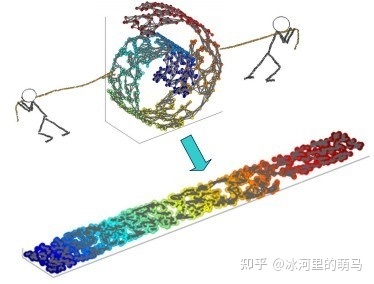


图2.2.1 流形降维过程图

实现步骤：

1.找到数据点的k个近邻点

2.由每个近邻点构建改样本点的局部重建矩阵

3.由局部重建矩阵和近邻点计算重建后的样本点

### 2.3 T-随机邻近嵌入（T-SNE）

T-随机邻近嵌入，它是一种嵌入模型，能够将高维空间中的数据映射到低维空间中，并保留数据集的局部特性，当我们想对高维数据集进行分类，但又不清楚这个数据集有没有很好的可分性（同类之间间隔小、异类之间间隔大）时，可以通过T-SNE将数据投影到2维或3维空间中观察一下：如果在低维空间中具有可分性，则数据是可分的；如果在低维空间中不可分，则可能是因为数据集本身不可分，或者数据集中的数据不适合投影到低维空间。

T-SNE将数据点之间的相似度转化为条件概率，原始空间中数据点的相似度由高斯联合分布表示，嵌入空间中数据点的相似度由学生t分布表示。通过原始空间和嵌入空间的联合概率分布的KL散度（用于评估两个分布的相似度的指标，经常用于评估机器学习模型的好坏）来评估嵌入效果的好坏，即将有关KL散度的函数作为损失函数（loss function），通过梯度下降算法最小化损失函数，最终获得收敛结果。

**2.4 三种算法对比**

**2.4.1 PCA与T-SNE**

T-SNE的计算复杂度远高于PCA，同一个数据集，在PCA运算需要几分钟的情况下，T-SNE的运算时间可能是若干小时；PCA是数学技巧，而T-SNE则属于概率的范畴；相同的超参也可能导致每次的T-SNE展示的结果不同，PCA则不是。

**2.4.2 PCA与LLE**

PCA 降维的缺陷：高维空间各个样本之间存在一些线性关系，降维之后并没有保留这些关系。比如，在高维空间中，其最短路径并不是三维空间中的两点之间直线最短，而是曲面距离，而我们降维后，就变成了投影距离。与 PCA 不同的是，LLE 保留了高维空间的局部线性关系。LLE在降维之前，先训练出每个样本与其最近的k个样本的线性关系——weights，再把该线性关系套用在降维空间后的每个样本。

**2.4.3 PCA、LLE与T-SNE**

PCA和LLE，这两种降维的方法都有一个共同的特点：在进行降维时，都强调了降维后的相似的数据要尽可能地保持相似，但并没有说对于那些不相似的数据，要有多不相似这个问题，这就导致了在进行降维时，可能导致数据的重叠问题，导致在低维空间中一样很难进行区分。这时就需要另一种降维的方法——T-SNE。所谓T-SNE，其就是SNE的升级版，SNE同前面的算法一样，希望样本在高维空间中相似的数据点，到低维空间也相似，而T-SNE则是在SNE的基础上，要使得对于高维空间不相似的样本，在低维空间中也尽可能不相似。

### 2.5 数据降维原理

所谓降维，即用一组个数为 d 的向量 Zi 来代表个数为 D 的向量 Xi 所包含的有用信息，其中 d<D，通俗来讲，即将高维度下降至低维度；将高维数据下降为低维数据。通常，我们会发现大部分数据集的维度都会高达成百乃至上千，而经典的 MNIST数据集 ，其维度都是 64。但在实际应用中，我们所用到的有用信息却并不需要那么高的维度，而且每增加一维所需的样本个数呈指数级增长，这可能会直接带来极大的维数灾难;而数据降维就可以实现：使得数据集更易使用；确保变量之间彼此独立；降低算法计算运算成本。

去除噪音一旦我们能够正确处理这些信息，正确有效地进行降维，这将大大有助于减少计算量，进而提高机器运作效率。而数据降维，也常应用于文本处理、人脸识别、图片识别、自然语言处理等领域。

往往高维空间的数据会出现分布稀疏的情况，所以在降维处理的过程中，我们通常会做一些数据删减，这些数据包括了冗余的数据、无效信息、重复表达内容等。例如：现有一张 1024\*1024 的图，除去中心 50\*50 的区域其它位置均为零值，这些为零的信息就可以归为无用信息;而对于对称图形而言，对称部分的信息则可以归为重复信息。因此，大部分经典降维技术也是基于这一内容而展开，其中降维方法又分为线性和非线性降维，非线性降维又分为基于核函数和基于特征值的方法。

**2.6. K最邻近法（KNN）**

K最邻近法，是最简单的[机器学习算法](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E7%AE%97%E6%B3%95/18635836" \t "https://baike.baidu.com/item/%E9%82%BB%E8%BF%91%E7%AE%97%E6%B3%95/_blank)之一。该方法的思路非常简单直观：如果一个样本在特征空间中的K个最相似的样本中的大多数属于某一个类别，则该样本也属于这个类别。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

KNN算法的核心思想是，如果一个样本在特征空间中的K个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN方法在类别决策时，只与极少量的相邻样本有关。由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合。

实现步骤：

1.准备数据，对数据进行预处理。

2.计算测试样本点（也就是待分类点）到其他每个样本点的距离。

3.对每个距离进行排序，然后选择出距离最小的K个点。

4.对K个点所属的类别进行比较，根据[少数服从多数](https://baike.baidu.com/item/%E5%B0%91%E6%95%B0%E6%9C%8D%E4%BB%8E%E5%A4%9A%E6%95%B0/10350157" \t "https://baike.baidu.com/item/%E9%82%BB%E8%BF%91%E7%AE%97%E6%B3%95/_blank)的原则，将测试样本点归入在K个点中占比最高的那一类。

# 3 特征降维结果展示与分析

### 3.1缩小维度范围

读入手写数据的数据，显示出数据的大小为1797\*64维，数据量庞大，为方便后续的计算，首先画出累计方差贡献率曲线（图3.1.1）和降维后维度的学习曲线（图3.1.2），找最佳降维后维度的范围。

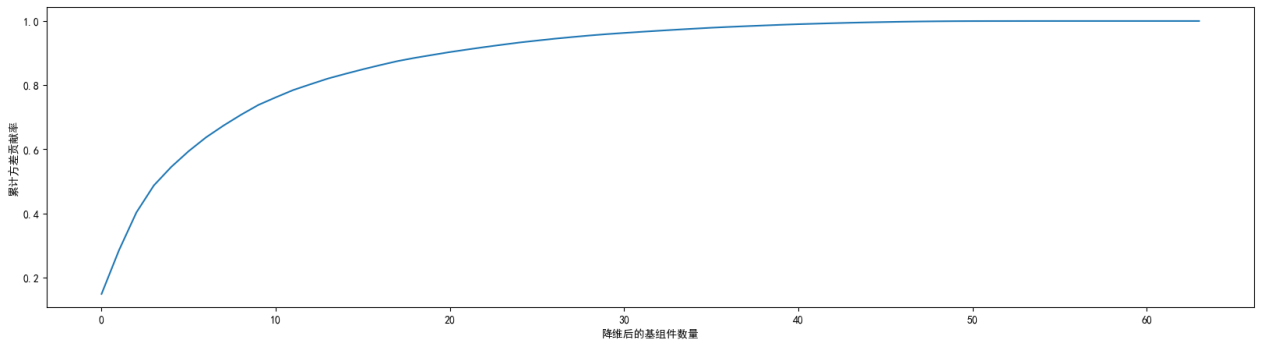


图3.1.1 累计方差贡献率曲线

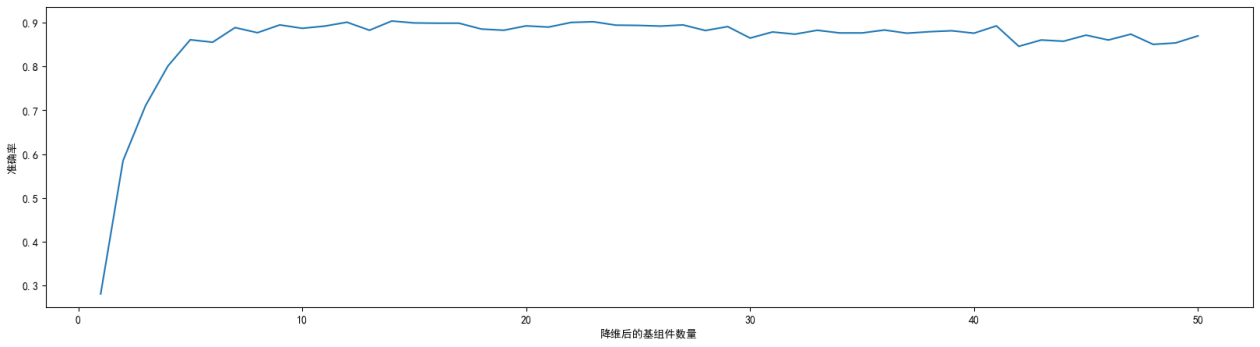


图3.1.2 降维后维度的学习曲线

从图3.1.1和图3.2.1中可以看出降维后的基组件数量在10以上时，累计方差贡献率能达到75%以上，准确率能达85%以上，为得到最佳维度，降低计算难度，继续细化学习曲线，画出基组件数量在10到20范围的学习曲线（图3.1.3）

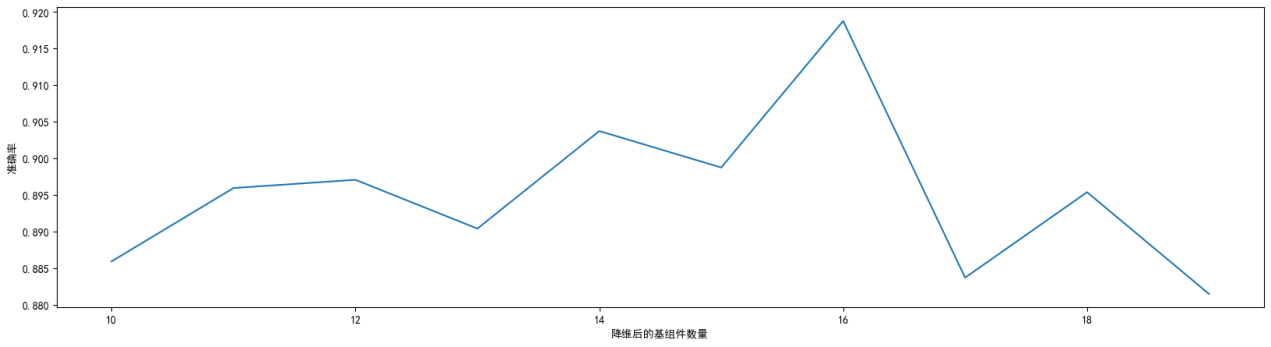


图3.1.3 细化后的学习曲线

由图3.1.3可以看出将原始数据64维特征变为 16 维特征时学习率最高，可以考虑将数据降维16维。

### 3.2主成分分析（PCA）

使用主成分分析（PCA）将数据64维特征变为 16 维特征， 使用随机森林和默认的KNN参数进行交叉验证，结果如下表：

表3.1.1 PCA随机交叉验证结果

|  |  |
| --- | --- |
| 随机森林 | KNN |
| 0.9282342927886104 | 0.9593980191891054 |

0.9282342927886104>0.9593980191891054所以在交叉验证环节使用KNN是否比随机森林好，改变K值，画出学习曲线，找出最优的K值。

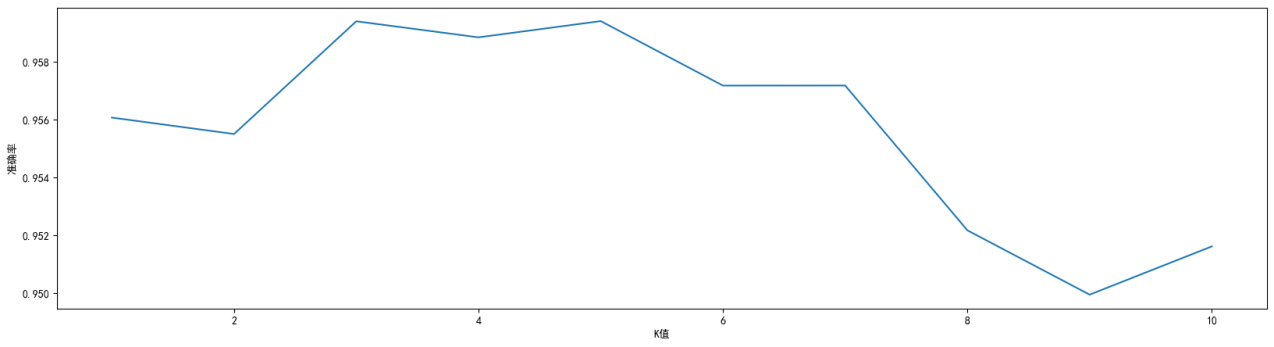


图3.2.1 KNN的K值学习曲线

由图3.2.1可知，K=5时最优，准确率为0.9593980191891054

### 3.3 T-随机邻近嵌入（T-SNE）

由于默认参数下的T-随机邻近嵌入（T-SNE）-的n\_components 小于4，即维数不能超过4，但由之前的实验可知T-随机邻近嵌入（T-SNE）在手写数据集上效果良好，所以将维数降为3维。使用KNN交叉验证，平均准确率为0.972742185082018；改变K值，画出学习曲线，找出最优的K值。结果如图3.3.1

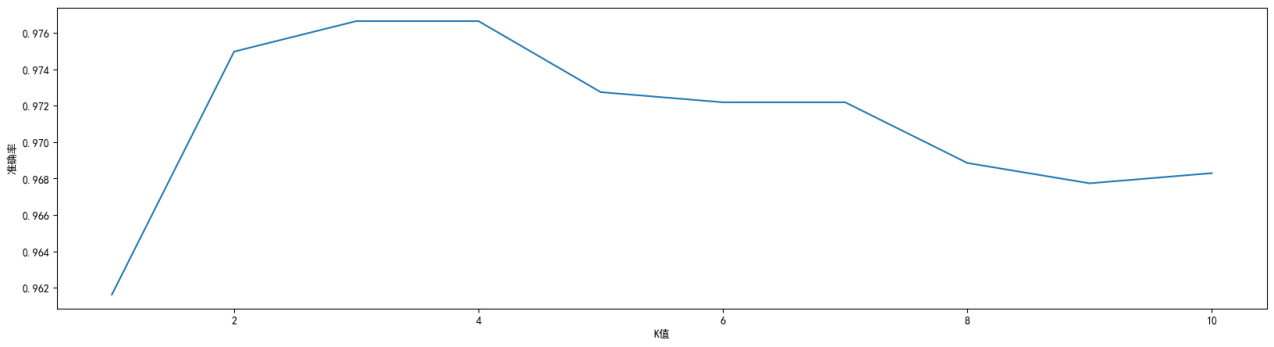


图3.3.1 KNN的K值学习曲线

由图3.3.1可知，K=3时最优，准确率为0.9766419065304859

### 3.4 局部线性嵌入（LLE）

局部线性嵌入（LLE）在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别 ，尝试局部线性嵌入（LLE）以下使用10作默认的邻居数，KNN交叉验证平均准确率为0.9026385020117612，为确定降维后的最佳维数和邻居数，输出不同维数和邻居数的准确率；由于单线程太慢，使用多线程加速，但结果是跑了2054秒，时间成本很高。结果为[(2, 18, 0.6561173011451562), (2, 19, 0.6845017022593625), (2, 17, 0.6895187248529868), (4, 5, 0.714535747446611), (2, 16, 0.7390560198081089), (2, 13, 0.7585267718972454), (2, 14, 0.7612999071494894), (2, 15, 0.7657582791705355), (3, 19, 0.7663030021665118), (3, 17, 0.7724295883627359),(3,18,0.7807520891364902), (4, 17, 0.8069204580625193), (4, 19, 0.8130578768183223), (3, 15, 0.8152615289384091), (3, 16, 0.8202940266171463), (2, 9, 0.8391999380996594), (2, 12, 0.8441906530485916), (4, 18, 0.8441999380996595), (3, 14, 0.8458650572578149), (2, 8, 0.8503265242958836)]

由此得最好的参数配置为 18维 特征，邻居6个，此时平均准确率为 0.9771959145775302，导入找出的最佳邻居数进行降维，KNN交叉验证平均准确率0.9771959145775302。改变K值，画出学习曲线，找出最优的K值。结果如图3.4.1

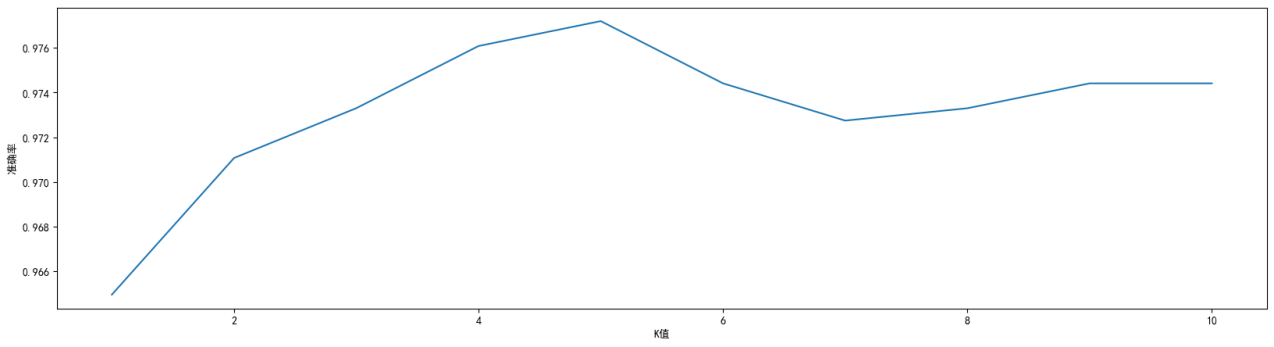


图3.4.1 KNN的K值学习曲线

由图3.3.1可知，K=5时最优，准确率为0.9771959145775302

### 3.5 结论

对比主成分分析（PCA）、局部线性嵌入（LLE）和 T-随机邻近嵌入（T-SNE）三个模型的准确率以及时间，结果如下表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 主成分分析（PCA） | 局部线性嵌入（LLE） | T-随机邻近嵌入（T-SNE） |
| 平均准确率 | 0.9593980191891054 | 0.972742185082018 | 0.9771959145775302 |
| 最优准确率 | 0.9593980191891054 | 0.9766419065304859 | 0.9771959145775302 |
| 花费时间 | 12.560771465301514 | 0.5395219326019287 | 2054.643949985504 |

表3.5.1 准确率对比

K值优化后，从准确率的结果为：局部线性嵌入（LLE）模型优于T-随机邻近嵌入（T-SNE）优于主成分分析（PCA），实际上局部线性嵌入（LLE）模型优于T-随机邻近嵌入（T-SNE）0.0017，提升不大，但T-随机邻近嵌入（T-SNE）的维度受限，只能是3维特征，主成分分析（PCA）(16维)和局部线性嵌入（LLE）(18维)会比较适合一点，数据集要保留的数据量更大， 其中局部线性嵌入（LLE）为了调成最优时间成本很高，花了2005秒，综合来看PCA会在准确率，数据量以及时间成本上更合适。

# 4 总结与展望

**4.1 总结**

关于手写数字识别的特征工程已经在很多行业领域发挥了极大的作用，也逐渐进入人们的生活。结合机器学习的研究发展，本文以手写数字特征为对象，分析对比局部线性嵌入（LLE）模型、T-随机邻近嵌入（T-SNE）、主成分分析（PCA）三个模型的特征提取效果，完成了如下几个方面的内容：

（1）首先在前言部分对关于手写数字识别的特征工程背景和研究现状等概况进行了全面具体的介绍，

2）关于局部线性嵌入（LLE）模型、T-随机邻近嵌入（T-SNE）、主成分分析（PCA）相关概念，算法步骤，区别等进行了全面的概述。

3）深入研究了局部线性嵌入（LLE）模型、T-随机邻近嵌入（T-SNE）、主成分分析（PCA）特征提取的准确率以及所用时间，并通过对比实验，从各方面分析了最合适的数据降维方法，并用KNN进行交叉验证。

### **4.2** 展望

由于实验条件和时间的问题，项目仍有不足之处，仍需做进一步的提高和完善，在本文的基础

上，依然存在一些问题，值得深入研究：

1）现实生活中，其实很多手写字迹都有自己的特点，特征的选择还需要进一步深入实践研究。

2）在选择降维模型时，没有尝试更多其他模型，或是使用不同的方法进行验证，还需要设计方案，并进行试验验证，有待完善与提高。

3）本论文的工作只是基于一组数据上的推导，功能上虽然可以满足降维要求，但是由于经验的缺乏，也没有参与到实际应用中去，现实中效果如何，还有待考查和检验。

**参考文献**

[1]郭科君. 面向高维复杂数据的特征降维方法研究[D].西南财经大学,2022.

[2]邓力珲,曹丽,张俊杰.基于t分布随机近邻嵌入的测井数据降维方法研究[J].合肥工业大学学报(自然科学版),2022,45(04):549-553+569.

[3]徐永辉.基于LLE-PCA的社交网络数据二重降维方法研究[J].现代电子技术,2022,45(10):69-74.

[4]郭依正,倪红军.基于LLE特征降维及改进SVM的肝脏图像识别[J].实验室研究与探索,2022,41(06):67-70.

[5]张婷婷. 随机森林的可视化解释与分析[D].电子科技大学,2021.

[6]黄光华,殷锋,冯九林.一种交叉验证和距离加权方法改进的KNN算法研究[J].西南民族大学学报(自然科学版),2020,46(02):172-177.

[7]吕兵,王华珍.基于随机森林的高维数据可视化[J].计算机应用,2014,34(06):1613-1617+1644.

[8]鞠玲. 局部线性嵌入算法及应用研究[D].扬州大学,2021.

[9]马宇. 基于高维空间的非线性降维的局部线性嵌入LLE方法[D].西南交通大学,2017.

[10]刘林林. 流形学习数据降维方法的研究[D].北京工业大学,2018.

**附录**

项目源代码：

from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier as RFC

from sklearn.neighbors import KNeighborsClassifier as KNN

from sklearn.model\_selection import cross\_val\_score

from sklearn import datasets, decomposition,manifold

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import time

import warnings

plt.rcParams['font.family'] = 'SimHei' ### # matplotlib其实是不支持显示中文的 显示中文需要一行代码设置字体

plt.rcParams['axes.unicode\_minus'] = False

warnings.filterwarnings ('ignore') ### 忽略版本警告

# # 读取并查看数据

# data = pd.read\_csv(r"digit recognizor.csv")

# X = data.iloc[:,1:]

# y = data.iloc[:,0]

# print(X.shape)

#加载数据，显示数据

digits = datasets.load\_digits()

X = digits.data

y = digits.target

print (X.shape,y.shape)

# 画累计方差贡献率曲线，找最佳降维后维度的范围

pca\_line = PCA().fit(X)

plt.figure(figsize=[20,5])

plt.plot(np.cumsum(pca\_line.explained\_variance\_ratio\_))

plt.xlabel("降维后的基组件数量")

plt.ylabel("累计方差贡献率")

plt.show()

#降维后维度的学习曲线，继续缩小最佳维度的范围

time\_start = time.time()

score = []

for i in range(1,51):

X\_pca = PCA(i).fit\_transform(X)

once = cross\_val\_score(RFC(n\_estimators=10,random\_state=0),X\_pca,y,cv=5).mean()

score.append(once)

plt.figure(figsize=[20,5])

plt.plot(range(1,51),score)

plt.xlabel("降维后的基组件数量")

plt.ylabel("准确率")

plt.show()

time\_end = time.time()

time\_sum = time\_end - time\_start

print(time\_sum)

#细化学习曲线，找出降维后的最佳维度

score = []

for i in range(10,20):

X\_pca = PCA(i).fit\_transform(X)

once = cross\_val\_score(RFC(n\_estimators=10,random\_state=0),X\_pca,y,cv=5).mean()

score.append(once)

plt.figure(figsize=[20,5])

plt.xlabel("降维后的基组件数量")

plt.ylabel("准确率")

plt.plot(range(10,20),score)

plt.show()

#导入找出的最佳维度进行降维，查看模型效果

X\_pca = PCA(16).fit\_transform(X) # 64 列特征变为 16 列特征

print(cross\_val\_score(RFC(n\_estimators=100,random\_state=0),X\_pca,y,cv=5).mean()) # 使用随机森林进行交叉验证

# 使用默认的KNN参数，看到在交叉验证环节使用KNN是否比随机森林好

print(cross\_val\_score(KNN(),X\_pca,y,cv=5).mean()) # 结果是KNN好

# KNN的K值学习曲线

time\_start = time.time()

score = []

for i in range(10):

X\_dr = PCA(16).fit\_transform(X)

once = cross\_val\_score(KNN(i+1),X\_pca,y,cv=5).mean()

score.append(once)

plt.figure(figsize=[20,5])

plt.xlabel("K值")

plt.ylabel("准确率")

plt.plot(range(1,11),score)

plt.show()

time\_end = time.time()

time\_sum = time\_end - time\_start

print(time\_sum)

#K值优化后的交叉验证

print(cross\_val\_score(KNN(5),X\_pca,y,cv=5).mean())

# 由之前的实验可知t-SNE在手写数据集上效果良好

# 默认参数下的t-SNE - n\_components 小于4，维度不能超过4

tsne = manifold.TSNE(n\_components=3, init='pca', random\_state=0, perplexity=10)

X\_tsne = tsne.fit\_transform(X)

print (X\_tsne.shape)

print(cross\_val\_score(KNN(),X\_tsne,y,cv=5).mean())

# KNN的K值学习曲线

time\_start = time.time()

score = []

for i in range(10):

once = cross\_val\_score(KNN(i+1),X\_tsne,y,cv=5).mean()

score.append(once)

plt.figure(figsize=[20,5])

plt.xlabel("K值")

plt.ylabel("准确率")

plt.plot(range(1,11),score)

plt.show()

time\_end = time.time()

time\_sum = time\_end - time\_start

print(time\_sum)

#K值优化后的交叉验证

print(cross\_val\_score(KNN(3),X\_tsne,y,cv=5).mean())

# 尝试LLE以下使用10作默认的邻居数

clf = manifold.LocallyLinearEmbedding(n\_neighbors=10, n\_components=2,method='standard')

X\_lle = clf.fit\_transform(X)

print(cross\_val\_score(KNN(),X\_lle,y,cv=5).mean())

# 画学习曲线，找出降维后的最佳维数和邻居数

# 单线程太慢，下面使用多线程加速

# best\_list = [] #(维数，邻居数，交叉验证的准确率)

# for i in range(2,64): # 维数

# score = []

# for j in range(3, 20): # 邻居数

# clf = manifold.LocallyLinearEmbedding(n\_neighbors=j, n\_components=i,method='standard')

# X\_lle = clf.fit\_transform(X)

# once = cross\_val\_score(KNN(),X\_lle,y,cv=5).mean()

# score.append(once)

# best\_list.append((i,score.index(max(score)),max(score)))

# print(best\_list)

# 多线程参数调优

# 多线程判断求解

import threading

import queue

from tqdm import tqdm

time\_start = time.time()

g = queue.Queue()

result\_g = queue.Queue()

count = 0

for i in range(2,64): # 维数

for j in range(3, 20): # 邻居数

g.put((i,j))

count += 1

def temp\_g():

while True:

if g.qsize() != 0:

elem = g.get()

i = elem[0]

j = elem[1]

clf = manifold.LocallyLinearEmbedding(n\_neighbors=j, n\_components=i,method='standard')

X\_lle = clf.fit\_transform(X)

once = cross\_val\_score(KNN(),X\_lle,y,cv=5).mean()

result\_g.put((i, j, once))

else:

break

threads = []

for i in tqdm(range(g.qsize())):

t = threading.Thread(target=temp\_g)

threads.append(t)

t.start()

print('主程序运行中...')

# 等待所有线程任务结束。

for t in threads:

t.join()

print("所有线程任务完成")

# 栈反解数据列表

result\_list = []

for i in range(result\_g.qsize()):

j = result\_g.get()

result\_list.append(j)

result\_list.sort(key=lambda x:x[2]) # 乱序排序

print(result\_list[0:20])

time\_end = time.time()

time\_sum = time\_end - time\_start

print(time\_sum) # 结果是跑了2054秒，时间成本很高

print(result\_list[-1]) #最好的参数配置 维数 邻居数 准确率

#导入找出的最佳邻居数进行降维，查看模型效果

clf = manifold.LocallyLinearEmbedding(n\_neighbors=6, n\_components=18,method='standard')

X\_lle = clf.fit\_transform(X)

print(cross\_val\_score(KNN(),X\_lle,y,cv=5).mean()) # 较未调参好了

# KNN的K值学习曲线

time\_start = time.time()

score = []

for i in range(10):

once = cross\_val\_score(KNN(i+1),X\_lle,y,cv=5).mean()

score.append(once)

plt.figure(figsize=[20,5])

plt.xlabel("K值")

plt.ylabel("准确率")

plt.plot(range(1,11),score)

plt.show()

time\_end = time.time()

time\_sum = time\_end - time\_start

print(time\_sum)

#K值优化后的交叉验证

print(cross\_val\_score(KNN(5),X\_lle,y,cv=5).mean())